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

Análise da Complexidade de Espaço para um Algoritmo de $K_{(1)}$-Validade

Bruno Azevedo Vilela

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

Orientadora
Prof.ª Dr.ª Cláudia Nalon

Brasília
2011

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

Coordenador: Prof. Dr. Marcus Vinicius Lamar

Banca examinadora composta por:

Prof.ª Dr.ª Cláudia Nalon (Orientadora) — CIC/UnB
Prof. Dr. Mauricio Ayala-Rincón — CIC/UnB
Prof.ª Dr.ª Maria Emília Machado Telles Walter — CIC/UnB

CIP — Catalogação Internacional na Publicação

Azevedo Vilela, Bruno.

Análise da Complexidade de Espaço para um Algoritmo de $K_{(1)}$-Validade / Bruno Azevedo Vilela. Brasília : UnB, 2011.
37 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2011.

1. Lógica Modal, 2. Complexidade de Algoritmos, 3. Validade

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

Análise da Complexidade de Espaço para um Algoritmo de $K_{(1)}$-Validade

Bruno Azevedo Vilela

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

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

Prof. Dr. Mauricio Ayala-Rincón
CIC/UnB

Prof.ª Dr.ª Maria Emília Machado Telles Walter
CIC/UnB

Prof. Dr. Marcus Vinicius Lamar
Coordenador do Bacharelado em Ciência da Computação

Brasília, 14 de Julho de 2011

Dedicatória

Dedico este trabalho ao meu pai Célio, à minha mãe Cristiana e à minha irmã Ana Carolina.

Agradecimentos

Agradeço a Deus, a minha família e aos meus amigos. Agradeço a minha orientadora Cláudia Nalon, por saber exigir e compreender quando necessário. Agradeço ao professor Guilherme de Albuquerque Pinto e ao professor Lineu Neto pelas melhores aulas que eu já assisti.

Resumo

Apresenta-se neste trabalho a análise da complexidade de espaço do cálculo de resolução para $K_{(1)}$ proposto por Nalon e Dixon. Prova-se que este algoritmo, para o problema de $K_{(1)}$-validade, tem complexidade de espaço de pior caso, pelo menos, exponencial, isto é, $f(n) = \Omega(2^n)$. Compara-se este resultado com a complexidade do algoritmo proposto por Ladner para o mesmo problema. Duas técnicas são sugeridas para uma melhora na complexidade de espaço. Prova-se, para isso, que é possível aplicar subsunção proposicional como parte do algoritmo baseado em resolução para $K_{(1)}$.

Palavras-chave: Lógica Modal, Complexidade de Algoritmos, Validade

Abstract

In this work, the analysis of the $K_{(1)}$'s resolution calculus space complexity proposed by Nalon and Dixon is done. It is proved that this algorithm, for the $K_{(1)}$-validity problem, has, at least, exponencial worst case space complexity that is $f(n) = \Omega(2^n)$. The result is compared with the algorithm's complexity proposed for the same problem by Ladner. For an improvement on the space's complexity, two technics are suggested. For this, it is proved that it is possible to apply propositional subsumption as part of the resolution based calculus for $K_{(1)}$.

Keywords: Modal Logic, Complexity of Algorithms, Validity

Sumário

Capítulo 1

Introdução

Um dos problemas mais famosos em teoria de complexidade é o de satisfatibilidade de uma fórmula lógica proposicional (**SAT**). Isto porque Cook [Coo71] provou que todos os problemas que pertencem a classe de complexidade **NP** podem ser reduzidos polinomialmente a **SAT** definindo com isso a classe de complexidade **NP-Difícil** e **NP-Completo**.

O problema de decidir se uma fórmula lógica é válida está relacionado com **SAT**, pois se uma fórmula φ é válida, então $\neg\varphi$ não é satisfatível [Tor04].

A lógica modal tem sido cada vez mais usada por ter a capacidade de modelar aspectos de sistemas computacionais complexos de uma forma bastante natural como o tempo, o conhecimento, etc [FHMV95]. Por exemplo, é usada para modelar uma série de aplicações de sistemas distribuídos [FHMV95]. A vantagem é que se consegue verificar as propriedades do sistema desenvolvido por meio de um provador de teoremas que analisa esta modelagem. Acontece que para modelar diferentes aspectos de uma situação talvez seja necessário a combinação de diferentes lógicas modais. Uma das combinações mais simples é a fusão de lógicas [GKWZ03], onde os componentes são independentemente axiomatizados. Isto significa que os cálculos de prova para a linguagem combinada podem potencialmente ser obtidos combinando os cálculos para cada linguagem. No entanto, é necessário garantir que as informações aplicáveis aos diferentes componentes sejam tratadas corretamente e trocadas entre estes componentes. Além disso, pode ser possível a utilização de cálculos com base em abordagens distintas para diferentes componentes, tornando a concepção de um cálculo completo para a linguagem combinada uma tarefa não trivial [ND07].

Para resolver esse problema, foi proposto um cálculo que pode ser utilizado em diferentes combinações de lógica modal [ND07], abrangendo um total de 16 famílias de lógicas modais. No entanto, não foi computada a complexidade de espaço ou de tempo para esses cálculos, afim de se verificar se são algoritmos eficientes.

Por isso, esta monografia tem o objetivo de apresentar a análise da complexidade de espaço do cálculo de resolução para $K_{(1)}$ comparando o resultado com a complexidade do algoritmo proposto por Richard E. Ladner, que é um algoritmo ótimo, em 1977 [LAD77].

1.1 Computabilidade

Nesta seção apresenta-se alguns conceitos de computabilidade [Sip97] importantes para o entendimento desta monografia.

1.1.1 Máquina de Turing

As máquinas de Turing foram propostas por Alan Turing[Tur36] em 1936, tendo como características principais possuir memória ilimitada de acesso irrestrito.

O modelo da máquina de Turing usa uma ou mais fitas de espaço infinito como memória ilimitada. Ela possui uma cabeça que pode ler e escrever símbolos e se mover sobre a fita. No estado inicial, a fita está preenchida somente pela cadeia de entrada e está em branco em todos os seus espaços de memória não preenchidos. Se a máquina precisar armazenar alguma informação, pode escrevê-la em algum espaço da fita. A máquina é capaz de ler somente o conteúdo apontado por sua cabeça. A máquina continua a computar até que esteja em um estado de aceitação ou de rejeição. Nesse caso, a saída é a palavra escrita na fita. Se ela não entrar em um desses estados ela continuará computando para sempre, sem nunca parar.

Definição 1 *Uma* ***máquina de Turing*** *é uma 7-upla* $(Q, \Sigma, \Gamma, \delta, q_0, q_{aceita}, q_{rejeita})$*, onde* Q*,* Σ *e* Γ *são todos conjuntos finitos:*

1. Q*, o conjunto de estados;*
2. Σ*, o alfabeto de entrada;*
3. Γ*, o alfabeto da fita, onde* $\Sigma \subseteq \Gamma$*;*
4. $\delta : Q - \{q_{aceita}, q_{rejeita}\} \times \Gamma \to Q \times \Gamma \times \{L, R\}$*, a função de transição;*
5. $q_0 \in Q$*, o estado inicial;*
6. $q_{aceita} \in Q$*, o estado de aceitação;*
7. $q_{rejeita} \in Q$*, o estado de rejeição.*

Uma máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{aceita}, q_{rejeita})$ funciona da seguinte forma. Primeiro M recebe uma entrada $w = w_1 w_2 ... w_n \in \Sigma^*$ nas células da sua fita; o restante da fita fica em branco, representado pelo símbolo $\sqcup \in \Gamma$ e $\sqcup \notin \Sigma$. A cabeça inicia na célula de w_1. A computação segue de acordo com as regras da função de transição δ, possivelmente provocando mudanças no seu estado atual, no conteúdo das células da fita e na posição da cabeça da máquina. A computação continua até que a máquina alcance algum estado de aceitação ou rejeição, onde a máquina para. Se nunca se chegar nesses estados, M nunca irá parar.

Definição 2 ***Configuração da Máquina de Turing****: Para um estado* q *e duas palavras* u *e* v *sobre o alfabeto da fita* Γ *escrevemos* uqv *para a configuração onde o estado atual é* q*, o conteúdo da fita é* u *e* v*, e a localização atual da cabeça é o primeiro símbolo de* v*. A fita contém apenas espaços em branco a partir do último símbolo de* v*.*

Definição 3 *Dada uma máquina de Turing* M *e uma palavra* $w \in \Sigma^*$*, dizemos que* M *aceita* w *se a sequência de configurações de* M *a partir de* $q_0 w \sqcup$ *alcança* q_{aceita}*.*

Definição 4 *Dada uma máquina de Turing* M*, a linguagem de* M *é:*

$$\mathcal{L}(M) = \{w \mid M \text{ aceita } w\}.$$

Definição 5 *Dizemos que uma máquina de Turing M* ***decide*** *$\mathcal{L}$, se M sempre para e $\mathcal{L}(M) = \mathcal{L}$.*

Definição 6 *Dizemos que uma máquina de Turing M* ***aceita*** *$\mathcal{L}$, se $\mathcal{L}(M) = \mathcal{L}$, mas M pode não parar quando $w \notin \mathcal{L}$.*

1.2 Complexidade de Algoritmo

Apresenta-se nesta seção os principais conceitos da teoria de complexidade [CLRS02, Pap94] que serão usados como fundamento para a análise da complexidade do cálculo estudado por este trabalho.

1.2.1 Notação Assintótica

Como o tempo e o espaço exato de execução de um algoritmo frequentemente é uma expressão complexa, em geral apenas o estimamos. Uma forma conveniente de estimativa chama-se análise assintótica que busca definir o tempo e o espaço gasto no algoritmo quando este é executado sobre uma entrada.

A notação assintótica representa a informação sobre a relação de crescimento assintótico entre duas funções, para um valor de entrada n.

Definição 7 ***Funções monotonicamente crescentes****: são funções que, a partir de certo ponto, são sempre não decrescentes, ou seja:*

$$\exists n_c(n_c \leq n \leq m \Rightarrow f(n) \leq f(m)).$$

Definição 8 ***Limitante Assintótico Superior****: Para uma dada função g, denotamos por $O(g)$ o conjunto de funções tal que:*

$$O(g) = \{f : \exists c > 0 \text{ e } \exists n_0 > 0 \text{ tal que } \forall n > n_0, 0 \leq f(n) \leq cg(n)\}.$$

Para declarar que $f(n) \in O(g(n))$, a sintaxe que normalmente se utiliza é $f = O(g)$. O Limitante Assintótico Superior é conhecido como Cota Superior e é o mesmo que dizer que a função f tem um crescimento assintótico menor ou igual que a função g.

Definição 9 ***Limite Assintótico Inferior****: Para uma dada função g, denotamos por $\Omega(g)$ o conjunto de funções tal que:*

$$\Omega(g) = \{f : \exists c > 0 \text{ e } \exists n_0 > 0 \text{ tal que } \forall n > n_0, 0 \leq cg(n) \leq f(n)\}.$$

Para declarar que $f(n) \in \Omega(g(n))$, a sintaxe normalmente utilizada é $f = \Omega(g)$. O Limitante Assintótico Inferior é conhecido como Cota Inferior e é o mesmo que dizer que a função f tem um crescimento assintótico maior ou igual que a função g.

Definição 10 ***Equivalência Assintótica****: Para uma dada função g, denotamos por $\Theta(g)$ o conjunto de funções tal que:*

$$\Theta(g)\{f : \exists c_1 > 0, \exists c_2 > 0 \text{ e } \exists n_0 > 0 \text{ tal que } \forall n > n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}.$$

Para declarar que $f(n) \in \Theta(g(n))$, a sintaxe normalmente utilizada é $f = \Theta(g)$. Equivalência Assintótica é o mesmo que dizer que ambas as funções tem o mesmo crescimento assintótico.

Definição 11 ***Algoritmo Ótimo****: Dado um problema P e um algoritmo A que resolve P, dizemos que A é um algoritmo ótimo de P, se $f_A = O(g(n))$ e $\Omega(g(n))$ é cota inferior para P, onde f_A é a função de custo de tempo ou de espaço de A.*

Intuitivamente, um algoritmo ótimo é aquele que apresenta a menor complexidade dentre todos aqueles que resolvem o mesmo problema.

1.2.2 Classes de Complexidade

Uma classe de complexidade é um conjunto de problemas, cuja pertinência é determinada por uma medida de complexidade, como o tempo de execução ou o espaço de memória utilizado.

Tempo e espaço são duas das mais importantes considerações quando buscamos soluções práticas para problemas computacionais.

Definição 12 ***Classe P****: Um problema A pertence à classe **P** se existe um algoritmo B que resolve o problema A, tal que, $f_B = O(n^k), k \in \mathbb{N}$, em uma máquina de Turing determinística.*

A classe **P** desempenha um papel importante na teoria de complexidade, pois a classe **P** é invariante para todos os modelos de computação que são polinomialmente equivalentes à máquina de Turing determinística de uma única fita e corresponde aproximadamente à classe de problemas que são solúveis realisticamente em um computador.

Definição 13 ***Classe NP****: Um problema A pertence à classe **NP** se existe um algoritmo B que resolve o problema A, tal que, $f_B = O(n^k), k \in \mathbb{N}$, em uma máquina de Turing não-determinística.*

Definição 14 ***Redutibilidade****: Uma linguagem $\mathcal{L}_1$ é redutível em tempo ou espaço polinomial a uma linguagem $\mathcal{L}_2$, o que se escreve como $\mathcal{L}_1 \leq_p \mathcal{L}_2$, se existe uma função computável de tempo ou espaço polinomial $f : \Sigma^* \longrightarrow \Sigma^*$ tal que, $\forall x \in \Sigma^*, x \in \mathcal{L}_1 \iff f(x) \in \mathcal{L}_2$.*

Intuitivamente, um problema A pode ser reduzido a outro problema B se qualquer instância de A pode ser transformada em tempo polinomial para uma instância de B, cuja solução fornece uma solução para a instância de A. As reduções de tempo polinomial nos permitem mostrar que um problema é pelo menos tão difícil quanto outro.

Outra medida de custo computacional é o espaço. A seguir apresentamos as definições das classes de complexidade referentes a espaço, que serão utilizadas neste trabalho.

Definição 15 ***Classe PSPACE****: Um problema A pertence à classe **PSPACE** se A possui uma cota superior de espaço $O(n^k), k \in \mathbb{N}$, ou seja, se existe um algoritmo que ocupa espaço polinomial para A.*

Definição 16 ***Classe NPSPACE****: Um problema A pertence à classe* ***NPSPACE*** *se A pode ser decidida por uma máquina de Turing não-determinística de complexidade de espaço de pior caso $O(n^k), k \in \mathbb{N}$.*

Pelo **teorema de Savitch** [Sip97] sabemos que $\mathbf{SPACE}(f) \subseteq \mathbf{NSPACE}(f^2)$, para qualquer função $f(n) \geq \log n$. Deste teorema, deduz-se que $\mathbf{PSPACE}(f) = \mathbf{NPSPACE}(f)$.

Capítulo 2

Lógica Modal Proposicional

Nesta seção, apresentaremos os principais conceitos de lógica modal[ND07, HC01, Che80] necessário ao entendimento deste trabalho..

2.1 Lógica Modal $K_{(1)}$

A classe da lógica modal mais básica é $K_{(1)}$, pois possui apenas o axioma da distribuição mais conhecido como axioma K e tem apenas um agente.

$$\textbf{Axioma K}: \Box(p \Rightarrow q) \Rightarrow (\Box p \Rightarrow \Box q)$$

2.1.1 Sintaxe

As fórmulas modais são construídas a partir dos seguintes símbolos lógicos:

- Símbolos proposicionais: $\mathcal{P} = \{p, q, r, ..., p_1, q_1, r_1, ...\}$;
- Operadores clássicos: $\{\neg, \vee, \wedge, \Rightarrow, \leftrightarrow\}$;
- Operadores modais: $\{\Box, \Diamond\}$;
- Constantes lógicas: $\{\textbf{true}, \textbf{false}\}$.

A diferença entre uma lógica modal e uma lógica clássica são os operadores modais, conhecidos como operador de necessidade ($\Box$) e operador de possibilidade ($\Diamond$).

Definição 17 *O conjunto de fórmulas bem-formadas de $K_{(1)}$, $WFF_{K_{(1)}}$, é dado pelo seguinte conjunto de regras:*

- $\boldsymbol{true} \in WFF_{K_{(1)}}$;
- *Se* $p \in \mathcal{P}$, *então* $p \in WFF_{K_{(1)}}$;
- *Se* $\varphi \in WFF_{K_{(1)}}$, *então* $\neg\varphi \in WFF_{K_{(1)}}$;
- *Se* $\varphi, \psi \in WFF_{K_{(1)}}$, *então* $(\varphi \wedge \psi) \in WFF_{K_{(1)}}$;
- *Se* $\varphi \in WFF_{K_{(1)}}$, *então* $\Box\varphi \in WFF_{K_{(1)}}$.

Os parênteses podem ser omitidos caso a leitura não seja ambígua. A ordem de precedência de operadores é:

$$\neg, \Box, \wedge, \vee$$

Definição 18 *Um **literal** é um símbolo proposicional ou sua negação.*

Definição 19 *Um **literal modal** é $\Box l$ ou $\neg\Box l$, onde l é um literal.*

2.1.2 Semântica

A semântica de $K_{(1)}$ é definida por meio de uma estrutura de Kripke.

Definição 20 *Uma **estrutura de Kripke** $\mathcal{M}$ é uma tripla $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \pi \rangle$, onde:*

- *$\mathcal{W}$ é o conjuntos dos mundos da estrutura de Kripke, com um mundo distinto s_0 e $\mathcal{W} \neq \emptyset$.*
- *$\mathcal{R}$ é o conjunto das relações entre os mundos.*
- *π é a função de avaliação dos símbolos em cada mundo.*

Os elementos do conjunto $\mathcal{R}$ são duplas (x_0, x_1), sendo $x_0, x_1 \in \mathcal{W}$, sendo por isso $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$.

A relação $\pi : \mathcal{W} \times \mathcal{P} \longrightarrow \{V, F\}$ é uma interpretação associada a estrutura de Kripke.

A satisfatibilidade de uma fórmula é definida em termos da relação $\vDash$. Nós escrevemos $(\mathcal{M}, w) \vDash \varphi$ para expressar que φ é satisfeita no mundo w na estrutura de Kripke $\mathcal{M}$.

Definição 21 *A relação de satisfatibilidade de uma fórmula é definida da seguinte forma:*

1. $(\mathcal{M}, w) \vDash true$*;*
2. $(\mathcal{M}, w) \vDash \varphi$*, se e somente se,* $\pi(w)(\varphi) = V$*, para* $\varphi \in \mathcal{P}$*;*
3. $(\mathcal{M}, w) \vDash \neg\varphi$*, se e somente se,* $(\mathcal{M}, w) \nvDash \varphi$*;*
4. $(\mathcal{M}, w) \vDash \varphi \wedge \psi$*, se e somente se,* $(\mathcal{M}, w) \vDash \varphi$ *e* $(\mathcal{M}, w) \vDash \psi$*;*
5. $(\mathcal{M}, w) \vDash \Box\varphi$*, se e somente se, para todo* w' *tal que* $w\mathcal{R}w'$*, nós temos que* $(\mathcal{M}, w') \vDash \varphi$*;*

As fórmulas **false**, $\varphi \vee \psi$, $\Diamond\varphi$ e $\varphi \Rightarrow \psi$ são introduzidas pelas abreviações usuais: $\neg$**true**, $\neg(\neg\varphi \wedge \neg\psi)$, $\neg\Box\neg\varphi$ e $\neg\varphi \vee \psi$, respectivamente.

Definição 22 *Uma fórmula φ é **satisfatível em uma estrutura de Kripke** $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \pi \rangle$ se é satisfeita no mundo s_0 de $\mathcal{M}$, ou seja, $(\mathcal{M}, s_0) \vDash \varphi$.*

Definição 23 *Uma fórmula φ é **satisfatível** se existe uma estrutura de Kripke $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \pi \rangle$ se é satisfeita no mundo s_0 de $\mathcal{M}$, ou seja, $(\mathcal{M}, s_0) \vDash \varphi$.*

Definição 24 *Uma fórmula φ é **válida** se é satisfatível em todas as estruturas de Kripke, ou seja $\forall \mathcal{M}, \mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \pi \rangle, (\mathcal{M}, s_0) \vDash \varphi$.*

2.2 Forma Normal Separada para $K_{(1)}$

Qualquer fórmula da linguagem $K_{(1)}$ pode ser transformada para a Forma Normal Separada para Lógica Modal (SNF_K) [ND06]. Uma fórmula em (SNF_K) é uma conjunção de cláusulas, que são satisfeitas em todos os estados, todos os mundos, tendo a forma geral:

$$\Box^* \bigwedge A_i$$

onde A_i é uma cláusula e $\Box^*$ é o operador universal, formalmente definido por $(\mathcal{M}, s) \models \Box^*\varphi$ se e somente se, $(\mathcal{M}, s) \models \varphi$ e para todos os $(s, s') \in \mathcal{R}$, $(\mathcal{M}, s') \models \Box^*\varphi$, onde $\mathcal{M}$ é uma estrutura de Kripke.

É também introduzida a constante **start** para representar o mundo inicial. Formalmente temos que, para uma estrutura de Kripke $\mathcal{M}$:

$$(\mathcal{M}, s) \vDash start, \text{ se e somente se, } s = s_0$$

As cláusulas estarão em uma destas formas:

Cláusulas iniciais:

$$start \Rightarrow \bigvee_{b=1}^{r} l_b$$

Cláusulas de literais:

$$true \Rightarrow \bigvee_{b=1}^{r} l_b$$

Cláusulas modais positivas:

$$l' \Rightarrow \Box l$$

Cláusulas modais negativas:

$$l' \Rightarrow \neg\Box l$$

onde l, l' e l_b são literais.

2.2.1 Transformação para (SNF_K)

A transformação de uma fórmula $\varphi \in K_{(1)}$ usa a técnica de renomeação [ND06]. Uma transformação de φ para (SNF_K) é baseada nas regras de tradução apresentadas abaixo, sendo x, y novos símbolos proposicionais.

$$\tau_0(\varphi) = (\Box^*(start \Rightarrow x)) \wedge \tau_1(\Box^*(x \Rightarrow \varphi))$$

$$\tau_1(\Box^*(x \Rightarrow \neg\neg A)) = \tau_1(\Box^*(x \Rightarrow A))$$

$$\tau_1(\Box^*(x \Rightarrow (A \wedge B))) = \tau_1(\Box^*(x \Rightarrow A)) \wedge \tau_1(\Box^*(x \Rightarrow B))$$

$$\tau_1(\Box^*(x \Rightarrow (A \Rightarrow B))) = \tau_1(\Box^*(x \Rightarrow \neg A \vee B))$$

$$\tau_1(\Box^*(x \Rightarrow \neg(A \wedge B))) = \tau_1(\Box^*(x \Rightarrow \neg A \vee \neg B))$$

$$\tau_1(\Box^*(x \Rightarrow \neg(A \Rightarrow B))) = \tau_1(\Box^*(x \Rightarrow A)) \wedge \tau_1(\Box^*(x \Rightarrow \neg B))$$

$$\tau_1(\Box^*(x \Rightarrow \neg(A \vee B))) = \tau_1(\Box^*(x \Rightarrow \neg A)) \wedge \tau_1(\Box^*(x \Rightarrow \neg B))$$

$$\tau_1(\Box^*(x \Rightarrow \Box A)) = \tau_1(\Box^*(x \Rightarrow \Box y)) \wedge \tau_1(\Box^*(y \Rightarrow A))$$

$$\tau_1(\Box^*(x \Rightarrow \neg\Box A)) = \tau_1(\Box^*(x \Rightarrow \neg\Box\neg y)) \wedge \tau_1(\Box^*(y \Rightarrow \neg A))$$

$$\tau_1(\Box^*(x \Rightarrow D \vee A)) = \tau_1(\Box^*(x \Rightarrow D \vee y)) \wedge \tau_1(\Box^*(y \Rightarrow A))$$

onde A e B não são literais. Além disso, não pode haver implicação ou negação de conjunção como operador principal.

$$\tau_1(\Box^*(x \Rightarrow D \vee (D' \Rightarrow D''))) = \tau_1(\Box^*(x \Rightarrow D \vee \neg D' \vee D''))$$

$$\tau_1(\Box^*(x \Rightarrow D \vee \neg(D' \wedge D''))) = \tau_1(\Box^*(x \Rightarrow D \vee \neg D' \vee \neg D''))$$

$$\tau_1(\Box^*(x \Rightarrow D)) = \tau_1(\Box^*(true \Rightarrow \neg x \vee D))$$

se D for disjunção de literais e

$$\tau_1(\Box^*(x \Rightarrow D)) = \tau_1(\Box^*(x \Rightarrow D))$$

se D for um literal modal.

Esta transformação preserva a satisfatibilidade da fórmula φ, como mostrado em [ND06].

Capítulo 3

Análise do Algoritmo de Resolução para $K_{(1)}$

Neste capítulo, iremos apresentar a análise de complexidade para o algoritmo de K_1-validade baseado em resolução [ND07, Rob65].

Dada uma fórmula da lógica modal φ decide-se se φ é uma fórmula válida em $K_{(1)}$, tendo como entrada $\neg\varphi \in SNF_K$, aplicando-se as regras de inferência até que $true \Rightarrow false$ ou $start \Rightarrow false$ sejam geradas (neste caso, φ é válida) ou não se possa aplicar mais nenhuma regra de inferência (neste caso, φ não é válida). As provas de terminação, corretude e completude deste cálculo e maiores detalhes se encontram em [ND07].

No que se segue, a apresentação em [ND07] foi adaptada para o caso de um único agente.

3.1 Regras de Inferência para $K_{(1)}$

Iremos apresentar as regras de inferência em dois grupos, as regras clássicas de resolução proposicional e as regras de resolução modal. Sejam $l, l', l_i, l_{i'}$ literais, $i \in \mathbb{N}$ e D, D' disjunção de literais.

Resolução de literais. Esta regra corresponde à resolução clássica aplicada à parte proposicional da lógica. Uma cláusula inicial pode ser resolvida com uma cláusula literal ou uma cláusula inicial (IRES1 e IRES2). Cláusulas de literais podem ser resolvidas entre si (LRES):

$$\text{[IRES1]}\ \frac{\Box^*(\mathbf{true} \Rightarrow D \vee l) \quad \Box^*(\mathbf{start} \Rightarrow D' \vee \neg l)}{\Box^*(\mathbf{start} \Rightarrow D \vee D')} \qquad \text{[IRES2]}\ \frac{\Box^*(\mathbf{start} \Rightarrow D \vee l) \quad \Box^*(\mathbf{start} \Rightarrow D' \vee \neg l)}{\Box^*(\mathbf{start} \Rightarrow D \vee D')} \qquad \text{[LRES]}\ \frac{\Box^*(\mathbf{true} \Rightarrow D \vee l) \quad \Box^*(\mathbf{true} \Rightarrow D' \vee \neg l)}{\Box^*(\mathbf{true} \Rightarrow D \vee D')}$$

Resolução Modal. Estas regras são aplicadas entre as cláusulas que se referem a um mesmo contexto, isto é, elas devem se referir ao mesmo agente. As regras de inferência modal são:

$$[\text{MRES}]\quad \frac{\begin{array}{rcl}\Box^*(l & \Rightarrow & \Box l_i)\\ \Box^*(l' & \Rightarrow & \neg\Box l_i)\end{array}}{\Box^*(\mathbf{true} \Rightarrow \neg l \vee \neg l')}$$

$$[\text{GEN2}]\quad \frac{\begin{array}{rcl}\Box^*(l'_1 & \Rightarrow & \Box l_1)\\ \Box^*(l'_2 & \Rightarrow & \Box\neg l_1)\\ \Box^*(l'_3 & \Rightarrow & \neg\Box\neg l_2)\end{array}}{\Box^*(\mathbf{true} \Rightarrow \neg l'_1 \vee \neg l'_2 \vee \neg l'_3)}$$

$$[\text{GEN1}]\quad \frac{\begin{array}{rcl}\Box^*(l'_1 & \Rightarrow & \Box\neg l_1)\\ & \vdots & \\ \Box^*(l'_m & \Rightarrow & \Box\neg l_m)\\ \Box^*(l' & \Rightarrow & \neg\Box\neg l)\\ \Box^*(\mathbf{true} & \Rightarrow & l_1 \vee \ldots \vee l_m \vee \neg l)\end{array}}{\Box^*(\mathbf{true} \Rightarrow \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l')}$$

$$[\text{GEN3}]\quad \frac{\begin{array}{rcl}\Box^*(l'_1 & \Rightarrow & \Box\neg l_1)\\ & \vdots & \\ \Box^*(l'_m & \Rightarrow & \Box\neg l_m)\\ \Box^*(l' & \Rightarrow & \neg\Box\neg l)\\ \Box^*(\mathbf{true} & \Rightarrow & l_1 \vee \ldots \vee l_m)\end{array}}{\Box^*(\mathbf{true} \Rightarrow \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l')}$$

Simplificação. Nós assumimos a simplificação padrão para a lógica clássica para deixar as cláusulas as mais simples possíveis. Por exemplo, $D \vee l \vee l$ deve ser simplificado e reescrito como $D \vee l$.

Abaixo, a aplicação de uma regra de inferência será indicado por $[c_1, c_2, \ldots, c_n, R]$, em que R é o nome da regra de inferência e cada c_i, $1 \leq i \leq n$, é o número da cláusula usada para aplicação da regra.

3.2 Análise de Pior Caso da Complexidade de Espaço do Algoritmo Proposto para $K_{(1)}$

Teorema 1 *A complexidade de espaço de pior caso $f(n)$ para o algoritmo de validade para $K_{(1)}$ [ND07] é pelo menos exponencial, isto é, $f(n) = \Omega(2^n)$, sendo $O(n)$ o número de cláusulas na entrada.*

Prova:

A prova será apresentada por meio de um exemplo que gera $\Omega(2^n)$ cláusulas com $O(n)$ cláusulas de entrada.

O cálculo decide se a fórmula modal φ é uma fórmula válida. Para isto, temos como entrada $\neg\varphi \in SNF_{(K)}$. Seja a transformação de $\neg\varphi$ dada pelo seguinte conjunto de cláusulas:

Nº	*Cláusula*
(1)	$\Box^*(\varphi_1 \Rightarrow \Box\neg\varphi_1)$
(2)	$\Box^*(\neg\varphi_1 \Rightarrow \Box\neg\varphi_1)$
(3)	$\Box^*(\varphi_2 \Rightarrow \Box\neg\varphi_2)$
(4)	$\Box^*(\neg\varphi_2 \Rightarrow \Box\neg\varphi_2)$
(5)	$\Box^*(\varphi_3 \Rightarrow \Box\neg\varphi_3)$
(6)	$\Box^*(\neg\varphi_3 \Rightarrow \Box\neg\varphi_3)$
$\vdots$	$\vdots$
$(2 \cdot n - 3)$	$\Box^*(\varphi_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
$(2 \cdot n - 2)$	$\Box^*(\neg\varphi_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
$(2 \cdot n - 1)$	$\Box^*(\varphi_n \Rightarrow \neg\Box\varphi_n)$
$(2 \cdot n)$	$\Box^*(\neg\varphi_n \Rightarrow \neg\Box\varphi_n)$
$(2 \cdot n + 1)$	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

Neste exemplo, estamos considerando a ocorrência de n símbolos proposicionais. A entrada é baseada no número de cláusulas. Como temos $2 \cdot n + 1$ *cláusulas este é o espaço ocupado pela entrada.*

Podemos aplicar $2^n - 1$ *vezes a regra* $[GEN1]$ *obtendo fórmulas diferentes que não podem ser simplificadas. O resultado será:*

Nº	*Cláusula*
$(2 \cdot n + 2)$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$
$(2 \cdot n + 3)$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \neg\varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$
$(2 \cdot n + 4)$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \neg\varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$
$\vdots$	$\vdots$
$(2 \cdot n + 2^n - 1)$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \neg\varphi_2 \vee \neg\varphi_3 \vee ... \vee \neg\varphi_{n-1} \vee \varphi_n)$
$(2 \cdot n + 2^n)$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \neg\varphi_2 \vee \neg\varphi_3 \vee ... \vee \neg\varphi_{n-1} \vee \neg\varphi_n)$

Com isso ocupamos na memória $2 \cdot n + 2^n$ *cláusulas, ou seja,* $f(n) = \Omega(2^n)$.
□

3.3 Comparação de Resultados

Para o problema de $K_{(1)}$-validade Ladner [LAD77] apresentou um algoritmo com complexidade de espaço $g(n) = O(n^2)$ e mostrou pela primeira vez que o problema pertence à classe **PSPACE**.

A comparação da complexidade de espaço do algoritmo proposto por Nalon e Dixon [ND07] e do algoritmo proposto por Ladner mostra que aquele algoritmo ocupa muito mais espaço que o ideal, principalmente por este ter complexidade de espaço polinomial e aquele ter complexidade de espaço exponencial.

No entanto, sugerem-se adiante duas técnicas que tentam otimizar o espaço para o cálculo apresentado em [ND07].A ideia consiste em propor uma ordem na aplicação das

regras de inferência e retirar da memória cláusulas desnecessárias diminuindo o espaço consumido.

3.4 Sugestões de Técnicas de Otimização de Espaço para o Cálculo de Resolução

Conforme visto no Teorema 1, o cálculo baseado em resolução para validade em $K_{(1)}$ pode gerar um grande número de cláusulas, mesmo tendo poucas cláusulas de entrada. Observa-se, por exemplo, que pode haver informação redundantes e que o número de cláusulas geradas pode ser influenciado pela ordem de aplicação das regras de inferência. No que segue faremos sugestões para lidar com ambos os problemas.

3.4.1 Subsunção

Para resolver a primeira questão levantada,ou seja, a de redundância de informações, usaremos subsunção que nos permite excluir da memória cláusulas que não colaboram para se encontrar uma contradição, na presença de outras cláusulas. Utilizaremos o seguinte resultado apresentado em [NCW95].

Teorema 2 ***Subsunção Proposicional****: Seja Σ um conjunto de cláusulas proposicionais. Sejam C e D cláusulas tais que $C, D \in \Sigma$ e $C \models D$. Σ é satisfatível, se e somente se, $\Sigma - \{D\}$ é satisfatível.*

Mostra-se abaixo que é possível aplicar subsunção proposicional em $K_{(1)}$.

Lema 1 *Seja $\Sigma \subseteq SNF_K$ um conjunto de cláusulas. Sejam C e D cláusulas literais tais que $C, D \in \Sigma$ e $C \models D$. Σ é satisfatível, se $\Sigma - \{D\}$ for satisfatível.*

Prova:
Como C e D são cláusulas literais quaisquer, então:

$$D = true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n$$
$$C = true \Rightarrow \psi_1 \vee \psi_2 \vee \psi_3 \vee ... \vee \psi_{m-1} \vee \psi_m$$

Como pela definição do problema $C \models D$ então $C \subseteq D$, portanto $m \leq n$.

Tem-se por hipótese que Σ é satisfatível. Para facilitar a compreensão da prova se separará a demonstração em casos:

1. *Não se consegue aplicar nenhuma regra de resolução em D, ou se consegue aplicar apenas regras de resolução proposicional em D, ou seja, [IRES1], [IRES2] ou [LRES]:*

   Neste caso se pode aplicar o teorema de subsunção proposicional, sendo assim $\Sigma - \{D\}$ é satisfatível.

2. *Consegue-se aplicar a regra de resolução modal [GEN3] em D, mas a cláusula gerada pela aplicação da regra D_1 não se consegue aplicar a regra [GEN1] ou [GEN3]:*

   Neste caso o fato de se poder aplicar [GEN3] em D necessita que as seguintes cláusulas necessariamente tem de pertencer a Σ:

Nº	*Cláusula*
(1)	$\Box^*(l_1 \Rightarrow \Box\neg\varphi_1)$
(2)	$\Box^*(l_2 \Rightarrow \Box\neg\varphi_2)$
(3)	$\Box^*(l_3 \Rightarrow \Box\neg\varphi_3)$
$\vdots$	$\vdots$
$(n-1)$	$\Box^*(l_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
(n)	$\Box^*(l_n \Rightarrow \Box\neg\varphi_n)$
$(n+1)$	$\Box^*(l_{n+1} \Rightarrow \neg\Box k)$

   Aplicando a regra $[D, 1, 2, \ldots, n, n+1, GEN3]$ geraremos uma nova cláusula:

$$D_1 = true \Rightarrow \neg l_1 \vee \neg l_2 \vee \neg l_3 \vee ... \vee \neg l_{n-1} \vee \neg l_n \vee \neg l_{n+1}$$

e definimos:

$$\Sigma^1 = \Sigma \cup D_1$$

   No entanto, como a cláusula $C \subseteq D$, então podemos aplicar a regra $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, i_{m+1}, GEN3]$, sendo $1 \leq i_j \leq n$, gerando como resultado uma nova cláusula:

$$C_1 = true \Rightarrow \neg t_1 \vee \neg t_2 \vee \neg t_3 \vee ... \vee \neg t_{m-1} \vee \neg t_m \vee \neg t_{m+1} =$$

$$true \Rightarrow \neg l_{i_1} \vee \neg l_{i_2} \vee \neg l_{i_3} \vee ... \vee \neg l_{i_{m-1}} \vee \neg l_{i_m} \vee \neg l_{i_{m+1}} \subset D_1$$

e definimos

$$\Sigma^1 = \Sigma^1 \cup C_1$$

   Como $C_1 \subset D_1$ tem-se que $C_1 \models D_1$.

   Como não se consegue aplicar mais nenhuma regra de resolução em D e talvez apenas regras proposicionais em D_1 vamos para o caso 1 da nossa demonstração. Com isso mostramos que $\Sigma^1 - \{D\} - \{D_1\} = \Sigma - \{D\} \cup \{C_1\}$ é satisfatível.

   Como C_1 é resultado da aplicação de $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, i_{m+1}, GEN3]$, sendo $1 \leq i_j \leq n$, então $\Sigma - \{D\}$ é satisfatível.

3. *Consegue-se aplicar a regra de resolução modal [GEN1] em D, mas a cláusula gerada pela aplicação da regra D_1 não se consegue aplicar a regra [GEN1] ou [GEN3] :*

   Neste caso o fato de se poder aplicar [GEN1] em D faz com que as seguintes cláusulas necessariamente tem de pertencer a Σ:

Nº	*Cláusula*
(1)	$\Box^*(l_1 \Rightarrow \Box\neg\varphi_1)$
(2)	$\Box^*(l_2 \Rightarrow \Box\neg\varphi_2)$
(3)	$\Box^*(l_3 \Rightarrow \Box\neg\varphi_3)$
$\vdots$	$\vdots$
$(n-1)$	$\Box^*(l_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
(n)	$\Box^*(l_n \Rightarrow \neg\Box\varphi_n)$

   Aplicando a regra $[D, 1, 2, \ldots, n, GEN1]$ geraremos uma nova cláusula:

$$D_1 = true \Rightarrow \neg l_1 \vee \neg l_2 \vee \neg l_3 \vee ... \vee \neg l_{n-1} \vee \neg l_n$$

e definimos:

$$\Sigma^1 = \Sigma \cup D_1$$

   No entanto, como a cláusula $C \subseteq D$, então podemos aplicar a regra [GEN1] ou [GEN3] na cláusula C gerando como resultado uma nova cláusula:

   (a) *Se $\varphi_n \in C$, então se pode aplicar a regra $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, GEN1]$, sendo $1 \leq i_j \leq n$, gerando como resultado uma nova cláusula:*

$$C_1 = true \Rightarrow \neg t_1 \vee \neg t_2 \vee \neg t_3 \vee ... \vee \neg t_{m-1} \vee \neg t_m =$$

$$true \Rightarrow \neg l_{i_1} \vee \neg l_{i_2} \vee \neg l_{i_3} \vee ... \vee \neg l_{i_{m-1}} \vee \neg l_{i_m} \subset D_1$$

Sendo então:

$$\Sigma^1 = \Sigma^1 \cup C_1$$

   Como $C_1 \subseteq D_1$ tem-se que $C_1 \models D_1$.
   Como não se consegue aplicar mais nenhuma regra de resolução em D e talvez apenas regras proposicionais em D_1 vai-se para o caso 1 da nossa demonstração. Com isso mostramos que $\Sigma^1 - \{D\} - \{D_1\} = \Sigma - \{D\} \cup \{C_1\}$ é satisfatível.
   Como C_1 e resultado da aplicação de $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, GEN1]$, então $\Sigma - \{D\}$ é satisfatível.

(b) *Se $\varphi_n \notin C$, então pode-se aplicar a regra $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, GEN3]$ gerando como resultado uma nova cláusula:*

$$C_1 = true \Rightarrow \neg t_1 \vee \neg t_2 \vee \neg t_3 \vee ... \vee \neg t_{m-1} \vee \neg t_m$$

$$= true \Rightarrow \neg l_{i_1} \vee \neg l_{i_2} \vee \neg l_{i_3} \vee ... \vee \neg l_{i_{m-1}} \vee \neg l_{i_m} \subset D_1$$

e definimos:

$$\Sigma^1 = \Sigma^1 \cup C_1$$

Como $C_1 \subseteq D_1$ tem-se que $C_1 \models D_1$ e $m \leq n$.
Como não se consegue aplicar mais nenhuma regra de resolução em D e talvez apenas regras proposicionais em D_1 vai-se para o caso 1 da nossa demonstração. Com isso mostra-se que $\Sigma^1 - \{D\} - \{D_1\} = \Sigma - \{D\} \cup \{C_1\}$ é satisfatível. Como C_1 e resultado da aplicação de $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, GEN3]$, então $\Sigma - \{D\}$ é satisfatível.

4. *Consegue-se aplicar a regra de resolução modal [GEN1] ou [GEN3] em D, a cláusula gerada pela aplicação da regra D_1 consegue-se aplicar a regra [GEN1] ou [GEN3] :*

   A prova é idêntica a demonstração 2 e 3 até o final, exceto que será possível aplicar as regras [GEN1] ou [GEN3] em D_1, mas não se pode aplicar nenhuma regra em D com isso vai-se para o caso 1 da demonstração, sendo $\Sigma^2 = \Sigma^1 - \{D\} = \Sigma - \{D\} \cup \{D_1\} \cup \{C_1\}$ é satisfatível.

   As cláusulas D_1 e C_1 têm as mesmas hipóteses de entrada do problema sendo $D = D_1$ e $C = C_1$. Aplica-se as regras modais [GEN1] ou [GEN3].Como é garantido a terminação do cálculo chegará um momento em que mais nenhuma regra de resolução modal poderá ser aplicada e se chegará no resultado que $\Sigma - \{D\}$ é satisfatível.

As regras [MRES] e [GEN2] são triviais visto que não operam sobre cláusulas literais, ou seja as cláusulas C e D não influenciam na aplicação dessas regras.
□

Lema 2 *Seja $\Sigma \subseteq SNF_K$ um conjunto de cláusulas. Sejam C e D cláusulas literais tais que $C, D \in \Sigma$ e $C \models D$. $\Sigma - \{D\}$ é satisfatível, se Σ for satisfatível.*

Prova:
Tem-se por hipótese que $\Phi = \Sigma - \{D\}$ é satisfatível. Para facilitar a compreensão da prova se separará a demonstração em casos:

1. *Não se consegue aplicar nenhuma regra de resolução em D, ou se consegue aplicar apenas regras de resolução proposicional em D, ou seja, [IRES1], [IRES2] ou [LRES]:*

   Neste caso se pode aplicar o teorema de subsunção proposicional, sendo assim Σ é satisfatível.

2. *Consegue-se aplicar a regra de resolução modal [GEN3] em D, mas a cláusula gerada pela aplicação da regra D_1 não se consegue aplicar a regra [GEN1] ou [GEN3] :*

   Neste caso o fato de se poder aplicar [GEN3] em D necessita que as seguintes cláusulas necessariamente tem de pertencer a Σ:

Nº	*Cláusula*
(1)	$\Box^*(l_1 \Rightarrow \Box\neg\varphi_1)$
(2)	$\Box^*(l_2 \Rightarrow \Box\neg\varphi_2)$
(3)	$\Box^*(l_3 \Rightarrow \Box\neg\varphi_3)$
$\vdots$	$\vdots$
$(n-1)$	$\Box^*(l_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
(n)	$\Box^*(l_n \Rightarrow \Box\neg\varphi_n)$
$(n+1)$	$\Box^*(l_{n+1} \Rightarrow \neg\Box k)$

Aplicando a regra $[D, 1, 2, \ldots, n, n+1, GEN3]$ *geraremos uma nova cláusula:*

$$D_1 = true \Rightarrow \neg l_1 \vee \neg l_2 \vee \neg l_3 \vee ... \vee \neg l_{n-1} \vee \neg l_n \vee \neg l_{n+1}$$

e definimos:

$$\Sigma^1 = \Sigma \cup D_1$$

No entanto, como a cláusula $C \subseteq D$, *então podemos aplicar a regra*

$$[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, i_{m+1}, GEN3]$$

sendo $1 \leq i_j \leq n$, *gerando como resultado uma nova cláusula:*

$$C_1 = true \Rightarrow \neg t_1 \vee \neg t_2 \vee \neg t_3 \vee ... \vee \neg t_{m-1} \vee \neg t_m \vee \neg t_{m+1} =$$

$$true \Rightarrow \neg l_{i_1} \vee \neg l_{i_2} \vee \neg l_{i_3} \vee ... \vee \neg l_{i_{m-1}} \vee \neg l_{i_m} \vee \neg l_{i_{m+1}} \subset D_1$$

e definimos

$$\Sigma^1 = \Sigma^1 \cup C_1$$

Como $C_1 \subset D_1$ *tem-se que* $C_1 \models D_1$.

Como não se consegue aplicar mais nenhuma regra de resolução em D e talvez apenas regras proposicionais em D_1 vamos para o caso 1 da nossa demonstração. Com isso mostramos que $\Sigma^1 = \Phi \cup \{D\} \cup \{D_1\} \cup \{C_1\}$ é satisfatível.

Como $\Sigma^1 = \Phi \cup \{D\} \cup \{D_1\} \cup \{C_1\}$ é satisfatível e $C_1 \models D_1$, concluímos pelo Lema 1 que $\Phi \cup \{D\} \cup \{C_1\}$ é satisfatível.

Como C_1 é resultado da aplicação de $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, i_{m+1}, GEN3]$, sendo $1 \leq i_j \leq n$, então $\Phi \cup \{D\} = \Sigma$ é satisfatível.

3. *Consegue-se aplicar a regra de resolução modal [GEN1] em D, mas a cláusula gerada pela aplicação da regra D_1 não se consegue aplicar a regra [GEN1] ou [GEN3] :*

   Neste caso o fato de se poder aplicar [GEN1] em D faz com que as seguintes cláusulas necessariamente tem de pertencer a Σ:

Nº	*Cláusula*
(1)	$\Box^*(l_1 \Rightarrow \Box\neg\varphi_1)$
(2)	$\Box^*(l_2 \Rightarrow \Box\neg\varphi_2)$
(3)	$\Box^*(l_3 \Rightarrow \Box\neg\varphi_3)$
$\vdots$	$\vdots$
$(n-1)$	$\Box^*(l_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
(n)	$\Box^*(l_n \Rightarrow \neg\Box\varphi_n)$

Aplicando a regra $[D, 1, 2, \ldots, n, GEN1]$ geraremos uma nova cláusula:

$$D_1 = true \Rightarrow \neg l_1 \vee \neg l_2 \vee \neg l_3 \vee ... \vee \neg l_{n-1} \vee \neg l_n$$

e definimos:

$$\Sigma^1 = \Sigma \cup D_1$$

No entanto, como a cláusula $C \subseteq D$, então podemos aplicar a regra [GEN1] ou [GEN3] na cláusula C gerando como resultado uma nova cláusula:

   (a) *Se $\varphi_n \in C$, então se pode aplicar a regra $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, GEN1]$, sendo $1 \leq i_j \leq n$, gerando como resultado uma nova cláusula:*

$$C_1 = true \Rightarrow \neg t_1 \vee \neg t_2 \vee \neg t_3 \vee ... \vee \neg t_{m-1} \vee \neg t_m =$$

$$true \Rightarrow \neg l_{i_1} \vee \neg l_{i_2} \vee \neg l_{i_3} \vee ... \vee \neg l_{i_{m-1}} \vee \neg l_{i_m} \subset D_1$$

Sendo então:

$$\Sigma^1 = \Sigma^1 \cup C_1$$

Como $C_1 \subseteq D_1$ *tem-se que* $C_1 \models D_1$.
Como não se consegue aplicar mais nenhuma regra de resolução em D *e talvez apenas regras proposicionais em* D_1 *vamos para o caso 1 da nossa demonstração. Com isso mostramos que* $\Sigma^1 = \Phi \cup \{D\} \cup \{D_1\} \cup \{C_1\}$ *é satisfatível.*
Como $\Sigma^1 = \Phi \cup \{D\} \cup \{D_1\} \cup \{C_1\}$ *é satisfatível e* $C_1 \models D_1$, *concluímos pelo Lema 1 que* $\Phi \cup \{D\} \cup \{C_1\}$ *é satisfatível.*
Como C_1 *é resultado da aplicação de* $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, i_{m+1}, GEN1]$, *sendo* $1 \leq i_j \leq n$, *então* $\Phi \cup \{D\} = \Sigma$ *é satisfatível.*

(b) *Se* $\varphi_n \notin C$, *então pode-se aplicar a regra* $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, GEN3]$ *gerando como resultado uma nova cláusula:*

$$C_1 = true \Rightarrow \neg t_1 \vee \neg t_2 \vee \neg t_3 \vee ... \vee \neg t_{m-1} \vee \neg t_m$$

$$= true \Rightarrow \neg l_{i_1} \vee \neg l_{i_2} \vee \neg l_{i_3} \vee ... \vee \neg l_{i_{m-1}} \vee \neg l_{i_m} \subset D_1$$

e definimos:

$$\Sigma^1 = \Sigma^1 \cup C_1$$

Como não se consegue aplicar mais nenhuma regra de resolução em D *e talvez apenas regras proposicionais em* D_1 *vamos para o caso 1 da nossa demonstração. Com isso mostramos que* $\Sigma^1 = \Phi \cup \{D\} \cup \{D_1\} \cup \{C_1\}$ *é satisfatível.*
Como $\Sigma^1 = \Phi \cup \{D\} \cup \{D_1\} \cup \{C_1\}$ *é satisfatível e* $C_1 \models D_1$, *concluímos pelo Lema 1 que* $\Phi \cup \{D\} \cup \{C_1\}$ *é satisfatível.*
Como C_1 *é resultado da aplicação de* $[C, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, i_{m+1}, GEN3]$, *sendo* $1 \leq i_j \leq n$, *então* $\Phi \cup \{D\} = \Sigma$ *é satisfatível.*

4. *Consegue-se aplicar a regra de resolução modal [GEN1] ou [GEN3] em* D, *a cláusula gerada pela aplicação da regra* D_1 *consegue-se aplicar a regra [GEN1] ou [GEN3] :*

   A prova é idêntica a demonstração 2 e 3 até o final, exceto que será possível aplicar as regras [GEN1] ou [GEN3] em D_1.

   As cláusulas D_1 *e* C_1 *têm as mesmas hipóteses de entrada do problema sendo* $D = D_1$ *e* $C = C_1$. *Aplica-se as regras modais [GEN1] ou [GEN3].Como é garantido a terminação do cálculo chegará um momento em que mais nenhuma regra de resolução modal poderá ser aplicada e se chegará no resultado que* $\Sigma^k = \Phi \cup \{D\} \cup \{D_1\} \cup \{C_1\} \cup \{D_2\} \cup \{C_2\} \cup \ldots \cup \{D_{k-1}\} \cup \{C_{k-1}\} \cup \{D_k\} \cup \{C_k\}$ *é satisfatível.*

   Como $\Sigma^k = \Phi \cup \{D\} \cup \{D_1\} \cup \{C_1\} \cup \{D_2\} \cup \{C_2\} \cup \ldots \cup \{D_{k-1}\} \cup \{C_{k-1}\} \cup \{D_k\} \cup \{C_k\}$ *é satisfatível e* $C_t \models D_t$, $1 \leq t \leq k$, *concluímos pelo Lema 1 que* $\Phi \cup \{D\} \cup \{C_1\} \cup \{C_2\} \cup \ldots \cup \{C_{k-1}\} \cup \{C_k\}$ *é satisfatível.*

   Como C_t *é resultado da aplicação de* $[C$ *ou* $C_{t-1}, i_1, i_2, \ldots, i_j, \ldots, i_{m-1}, i_m, i_{m+1}, GEN1$ *ou* $GEN3]$, *sendo* $1 \leq i_j \leq n$, *então* $\Phi \cup \{D\} = \Sigma$ *é satisfatível.*

As regras [MRES] e [GEN2] são triviais visto que não operam sobre cláusulas literais, ou seja as cláusulas C e D não influenciam na aplicação dessas regras.
□

Teorema 3 ***Subsunção Proposicional em*** $K_{(1)}$*: Seja* $\Sigma \subseteq SNF_K$ *um conjunto de cláusulas. Sejam* C *e* D *cláusulas literais tais que* $C, D \in \Sigma$ *e* $C \models D$*.* Σ *é satisfatível, se e somente se,* $\Sigma - \{D\}$ *é satisfatível.*

Prova:
Demonstração decorre do Lema 1 e do Lema 2.
□

Apresenta-se abaixo um exemplo que mostra como subsunção proposicional pode diminuir o número de cláusulas armazenadas.

Exemplo 1 *Seja os símbolos proposicionais* p *,* q *,* r *e* $\varphi \in SNF_K$*, apresentado abaixo:*

Nº	*Cláusula*
(1)	$\Box^*(true \Rightarrow p)$
(2)	$\Box^*(true \Rightarrow \neg q \vee r)$
(3)	$\Box^*(true \Rightarrow \neg p \vee \neg q \vee r)$
(4)	$\Box^*(r \Rightarrow \Box\neg p)$
(4)	$\Box^*(q \Rightarrow \neg\Box\neg q)$

A clausula $\Box^*(true \Rightarrow p)$ *elimina a possibilidade de ocorrência de todas as cláusulas abaixo por meio de subsunção:*

Nº	*Cláusula*
(1)	$\Box^*(true \Rightarrow p \vee q)$
(2)	$\Box^*(true \Rightarrow p \vee \neg q)$
(3)	$\Box^*(true \Rightarrow p \vee r)$
(4)	$\Box^*(true \Rightarrow p \vee \neg r)$
(5)	$\Box^*(true \Rightarrow p \vee q \vee r)$
(6)	$\Box^*(true \Rightarrow p \vee q \vee \neg r)$
(7)	$\Box^*(true \Rightarrow p \vee \neg q \vee r)$
(8)	$\Box^*(true \Rightarrow p \vee \neg q \vee \neg r)$

A clausula $\Box^*(true \Rightarrow q \vee r)$ *elimina a possibilidade de ocorrência de todas as cláusulas abaixo por meio de subsunção:*

Nº	*Cláusula*
(1)	$\Box^*(true \Rightarrow \neg q \vee r \vee p)$
(2)	$\Box^*(true \Rightarrow \neg q \vee r \vee \neg p)$

A clausula $\Box^*(true \Rightarrow \neg p \vee \neg q \vee r)$ *não elimina a possibilidade de ocorrência de cláusula alguma. Pois possui o maior número de literais em um cláusula.*

Com este exemplo, fica bastante claro que nosso objetivo para melhorar a complexidade de espaço do problema é tentar gerar cláusulas com poucos literais.

3.4.2 Definindo uma Ordem de Aplicação das Regras de Inferência

De acordo com o que foi apresentado, a melhor ordem de aplicação das regras de inferência é aquela que prioriza a geração de cláusulas com poucos literais. Por isso define-se a seguinte ordem:

1. MRES: Pois gera cláusulas que contém sempre dois literais.
2. GEN2: Pois gera cláusulas que contém sempre três literais.
3. IRES1, IRES2, LRES: Nada específico.
4. GEN1: Pois pode gerar cláusulas menores que GEN3.
5. GEN3: Nada específico.

Em relação as regras [GEN1] e [GEN3] é importante definir que sempre que houver mais de uma possibilidade de cláusulas geradas por uma mesma cláusula de literais ao se aplicar $[GEN1]$ ou $[GEN3]$, será gerada primeiro uma cláusula que tenha apenas um literal diferente de alguma cláusula contida na memória. Caso não haja, pode-se escolher uma cláusula qualquer. Abaixo se apresentará um exemplo disso:

Exemplo 2 *Considere o seguinte conjunto de cláusulas* φ_1 , φ_2 , φ_3, φ_4 *e* $\varphi \in SNF_K$, *que são literais.*

Nº	*Cláusula*
(1)	$\Box^*(\varphi_1 \Rightarrow \Box\neg\varphi_1)$
(2)	$\Box^*(\neg\varphi_1 \Rightarrow \Box\neg\varphi_1)$
(3)	$\Box^*(\varphi_2 \Rightarrow \Box\neg\varphi_2)$
(4)	$\Box^*(\neg\varphi_2 \Rightarrow \Box\neg\varphi_2)$
(5)	$\Box^*(\varphi_3 \Rightarrow \Box\neg\varphi_3)$
(6)	$\Box^*(\neg\varphi_3 \Rightarrow \Box\neg\varphi_3)$
(7)	$\Box^*(\varphi_4 \Rightarrow \neg\Box\varphi_4)$
(8)	$\Box^*(\neg\varphi_4 \Rightarrow \neg\Box\varphi_4)$
(9)	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4)$

Seguindo a ordem estabelecida acima vemos que só se pode aplicar a regra [GEN1].
No entanto [GEN1] aplicado a $true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4$ *pode gerar inúmeras cláusulas:*

Regra de Inferência	*Cláusula*
$([1,3,5,7,9,GEN1])$	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4)$
$([2,3,5,7,9,GEN1])$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4)$
$([1,4,5,7,9,GEN1])$	$\Box^*(true \Rightarrow \varphi_1 \vee \neg\varphi_2 \vee \varphi_3 \vee \varphi_4)$
$([1,3,6,7,9,GEN1])$	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \neg\varphi_3 \vee \varphi_4)$
$([1,3,5,8,9,GEN1])$	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \neg\varphi_4)$
$\vdots$	$\vdots$
$([2,4,6,7,9,GEN1])$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \neg\varphi_2 \vee \neg\varphi_3 \vee \varphi_4)$
$([2,4,6,8,9,GEN1])$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \neg\varphi_2 \vee \neg\varphi_3 \vee \neg\varphi_4)$

Qual dessas possibilidades gerar? Como definido na regra acima se escolhe primeiro sempre a cláusula que possuir apenas um literal diferente das cláusulas armazenadas em memória. Neste caso temos quatro possibilidades.

Regra de Inferência	*Cláusula*
$([2,3,5,7,9,GEN1])$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4)$
$([1,4,5,7,9,GEN1])$	$\Box^*(true \Rightarrow \varphi_1 \vee \neg\varphi_2 \vee \varphi_3 \vee \varphi_4)$
$([1,3,6,7,9,GEN1])$	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \neg\varphi_3 \vee \varphi_4)$
$([1,3,5,8,9,GEN1])$	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \neg\varphi_4)$

Pode-se escolher qualquer uma dessas quatro cláusulas.

É importante que haja esta seleção, pois esta cláusula gerada sempre poderá aplicar a regra de inferência [LRES] gerando uma cláusula com um número menor de literais que por subsunção elimina essas duas cláusulas.

Suponha que se tenha escolhido:

Regra de Inferência	*Cláusula*
$([2,3,5,7,9,GEN1])$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4)$

Com isso se pode aplicar as regras [LRES] e [GEN1], mas seguindo a ordem proposta se aplicará $[9,10,LRES]$*:*

Nº	*Cláusula*
(9)	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4)$
(10)	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4)$

Tendo como resultado:

Nº	*Cláusula*
(11)	$\Box^*(true \Rightarrow \varphi_2 \vee \varphi_3 \vee \varphi_4)$

Como sempre ao se aplicar uma regra se tenta realizar subsunção, vê-se que neste caso é possível eliminando da memória as regras:

Nº	*Cláusula*
(9)	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4)$
(10)	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4)$

Observa-se que nesse exemplo usamos as duas técnicas sugeridas juntas. Acredita-se que a aplicação das técnicas propostas faz com que a complexidade de espaço do cálculo [ND07] seja $O(n^2)$ igual ao do algoritmo proposto por Ladner [LAD77].

Iremos resolver de novo o exemplo usado para provar que o cálculo possui complexidade de espaço pelo menos $\Omega(2^n)$ e veremos o custo de espaço para este exemplo utilizando as duas técnicas sugeridas.

3.4.3 Exemplo Difícil Utilizando as Duas Técnicas Sugeridas.

Queremos saber se φ é uma fórmula válida. Para isto, temos como entrada para o algoritmo $\neg\varphi \in SNF_{(K)}$:

Nº	Cláusula
(1)	$\Box^*(\varphi_1 \Rightarrow \Box\neg\varphi_1)$
(2)	$\Box^*(\neg\varphi_1 \Rightarrow \Box\neg\varphi_1)$
(3)	$\Box^*(\varphi_2 \Rightarrow \Box\neg\varphi_2)$
(4)	$\Box^*(\neg\varphi_2 \Rightarrow \Box\neg\varphi_2)$
(5)	$\Box^*(\varphi_3 \Rightarrow \Box\neg\varphi_3)$
(6)	$\Box^*(\neg\varphi_3 \Rightarrow \Box\neg\varphi_3)$
$\vdots$	$\vdots$
$(2 \cdot n - 3)$	$\Box^*(\varphi_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
$(2 \cdot n - 2)$	$\Box^*(\neg\varphi_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
$(2 \cdot n - 1)$	$\Box^*(\varphi_n \Rightarrow \neg\Box\varphi_n)$
$(2 \cdot n)$	$\Box^*(\neg\varphi_n \Rightarrow \neg\Box\varphi_n)$
$(2 \cdot n + 1)$	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

Sendo $n \in \mathbb{N}$, se tem n símbolos proposicionais possíveis.

A entrada é baseada no número de cláusulas, como se têm $2 \cdot n + 1$ cláusulas este é o espaço ocupado pela entrada.

Pode-se aplicar somente a regra $[GEN1]$ podendo gerar $2^n - 1$ resultados diferentes, mas se tem de observar o critério de boa escolha. Com isso se tem a opção de gerar as seguintes cláusulas:

$$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$$

$$\Box^*(true \Rightarrow \varphi_1 \vee \neg\varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$$

$$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \neg\varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$$

$$\vdots$$

$$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \neg\varphi_{n-1} \vee \varphi_n)$$

$$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \neg\varphi_n)$$

Ou seja, apenas n opções. Escolhe-se a primeira opção:

Nº	Cláusula
$(2 \cdot n + 2)$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

Com isso pode-se aplicar as regras [LRES] e [GEN1], mas seguindo a ordem proposta aplica-se $[2 \cdot n + 1, 2 \cdot n + 2, LRES]$:

Nº	Cláusula
$(2 \cdot n + 1)$	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$
$(2 \cdot n + 2)$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

Tendo como resultado:

Nº	Cláusula
$(2 \cdot n + 3)$	$\Box^*(true \Rightarrow \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

Como sempre ao se aplicar uma regra tenta-se realizar subsunção, vê-se que neste caso é possível eliminando da memória as regras:

Nº	Cláusula
$(2 \cdot n + 1)$	$\Box^*(true \Rightarrow \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$
$(2 \cdot n + 2)$	$\Box^*(true \Rightarrow \neg\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

Tem-se portanto na memória:

Nº	Cláusula
(1)	$\Box^*(\varphi_1 \Rightarrow \Box\neg\varphi_1)$
(2)	$\Box^*(\neg\varphi_1 \Rightarrow \Box\neg\varphi_1)$
(3)	$\Box^*(\varphi_2 \Rightarrow \Box\neg\varphi_2)$
(4)	$\Box^*(\neg\varphi_2 \Rightarrow \Box\neg\varphi_2)$
(5)	$\Box^*(\varphi_3 \Rightarrow \Box\neg\varphi_3)$
(6)	$\Box^*(\neg\varphi_3 \Rightarrow \Box\neg\varphi_3)$
$\vdots$	$\vdots$
$(2 \cdot n - 3)$	$\Box^*(\varphi_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
$(2 \cdot n - 2)$	$\Box^*(\neg\varphi_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
$(2 \cdot n - 1)$	$\Box^*(\varphi_n \Rightarrow \neg\Box\varphi_n)$
$(2 \cdot n)$	$\Box^*(\neg\varphi_n \Rightarrow \neg\Box\varphi_n)$
$(2 \cdot n + 3)$	$\Box^*(true \Rightarrow \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

Pode-se aplicar as regras $[GEN1]$ ou $[GEN3]$. Seguindo a ordem de aplicação das regras estabelecidas se aplicará $[GEN1]$ podendo gerar várias cláusulas diferentes, mas se tem de observar o critério de boa escolha. Com isso se tem a opção de gerar as seguintes cláusulas:

$$\Box^*(true \Rightarrow \neg\varphi_2 \vee \varphi_3 \vee \varphi_4 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$$
$$\Box^*(true \Rightarrow \varphi_2 \vee \neg\varphi_3 \vee \varphi_4 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$$
$$\Box^*(true \Rightarrow \varphi_2 \vee \varphi_3 \vee \neg\varphi_4 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$$
$$\vdots$$
$$\Box^*(true \Rightarrow \varphi_2 \vee \varphi_3 \vee \varphi_4 \vee ... \vee \neg\varphi_{n-1} \vee \varphi_n)$$
$$\Box^*(true \Rightarrow \varphi_2 \vee \varphi_3 \vee \varphi_4 \vee ... \vee \varphi_{n-1} \vee \neg\varphi_n)$$

Ou seja, apenas $n - 1$ opções. Escolhe-se a primeira opção:

Nº	Cláusula
$(2 \cdot n + 4)$	$\Box^*(true \Rightarrow \neg\varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

Com isso se pode aplicar as regras [LRES] e [GEN1] e [GEN3], mas seguindo a ordem proposta se aplicará $[2 \cdot n + 3, 2 \cdot n + 4, LRES]$:

Nº	Cláusula
$(2 \cdot n + 3)$	$\Box^*(true \Rightarrow \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$
$(2 \cdot n + 4)$	$\Box^*(true \Rightarrow \neg\varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

Tendo como resultado:

Nº	Cláusula
$(2 \cdot n + 5)$	$\Box^*(true \Rightarrow \neg\varphi_3 \vee \varphi_4 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

Como sempre ao se aplicar uma regra tenta-se realizar subsunção, vê-se que neste caso é possível eliminando da memória as regras:

Nº	Cláusula
$(2 \cdot n + 3)$	$\Box^*(true \Rightarrow \varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$
$(2 \cdot n + 4)$	$\Box^*(true \Rightarrow \neg\varphi_2 \vee \varphi_3 \vee ... \vee \varphi_{n-1} \vee \varphi_n)$

$\vdots$

$\vdots$

Pode-se observar que se gera um ciclo que cada vez se vai eliminando um literal do conjunto de cláusula. Após aplicar mais $n - 3$ vezes este processo se terá o seguinte conjunto de cláusulas:

Nº	Cláusula
(1)	$\Box^*(\varphi_1 \Rightarrow \Box\neg\varphi_1)$
(2)	$\Box^*(\neg\varphi_1 \Rightarrow \Box\neg\varphi_1)$
(3)	$\Box^*(\varphi_2 \Rightarrow \Box\neg\varphi_2)$
(4)	$\Box^*(\neg\varphi_2 \Rightarrow \Box\neg\varphi_2)$
(5)	$\Box^*(\varphi_3 \Rightarrow \Box\neg\varphi_3)$
(6)	$\Box^*(\neg\varphi_3 \Rightarrow \Box\neg\varphi_3)$
$\vdots$	$\vdots$
$(2 \cdot n - 3)$	$\Box^*(\varphi_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
$(2 \cdot n - 2)$	$\Box^*(\neg\varphi_{n-1} \Rightarrow \Box\neg\varphi_{n-1})$
$(2 \cdot n - 1)$	$\Box^*(\varphi_n \Rightarrow \neg\Box\varphi_n)$
$(2 \cdot n)$	$\Box^*(\neg\varphi_n \Rightarrow \neg\Box\varphi_n)$
$(2 \cdot n + 2 \cdot (n-1))$	$\Box^*(true \Rightarrow \varphi_n)$

Pode-se aplicar as regras $[GEN1]$ ou $[GEN3]$. Seguindo a ordem de aplicação das regras estabelecidas se aplicará $[GEN1]$ podendo gerar vários resultados diferentes. Mas, seguindo a definição de boa escolha só se terá uma opção:

Nº	Cláusula
$(2{\cdot}n{+}2{\cdot}(n{-}1){+}1)$	$\Box^*(true \Rightarrow \neg\varphi_n)$

Com isso se pode aplicar as regras [LRES] e [GEN1] e [GEN3], mas seguindo a ordem proposta se aplicará $[2 \cdot n + 2 \cdot (n-1), 2 \cdot n + 2 \cdot (n-1) + 1, LRES]$:

N^o	Cláusula
$(2 \cdot n + 2 \cdot (n-1))$	$\Box^*(true \Rightarrow \varphi_n)$
$(2 \cdot n + 2 \cdot (n-1) + 1)$	$\Box^*(true \Rightarrow \neg\varphi_n)$

Tendo como resultado a cláusula vazia $\Box$.

Chega-se assim a uma contradição e prova-se que φ é uma fórmula válida.

Na análise deste exemplo, com este novo método, o espaço utilizado é assintoticamente constante, pois só tivemos que armazenar na memória, além da entrada, duas cláusulas. Isso não prova a eficiência da complexidade de espaço dessas técnicas, mas é um indicador de melhora.

Capítulo 4

Conclusão

Neste trabalho foi realizada uma análise de pior caso da complexidade de espaço para o algoritmo de $K_{(1)}$-Validade proposto por Nalon e Dixon [ND07]. Primeiramente, mostramos que este algoritmo tem complexidade de espaço pelo menos exponencial, isto é $\Omega(2^n)$. Além disso, comparou-se este resultado com o algoritmo proposto por Ladner [LAD77] que apresenta complexidade de espaço $O(n^2)$, notando-se que aquele apresenta uma complexidade de espaço alta comparativamente com este.

Foram sugeridas duas técnicas para otimização do espaço que se acredita melhorar a utilização da memória, devido ao último exemplo apresentado, apesar de não se ter provado isto. Foi provado que se pode aplicar subsunção proposicional para o cálculo para $K_{(1)}$ e foi proposta uma ordem para se aplicar as regras de inferência.

No segundo caso, após utilizar a ordenação sugerida e subsunção, o consumo de espaço para o exemplo apresentado no Teorema 1 foi reduzido. Acredita-se que, usando as técnicas sugeridas, pode-se reduzir a complexidade de espaço para o cálculo apresentado em [ND07] para $O(n^2)$,.

Para trabalhos futuros propõe-se a prova de complexidade de espaço para o algoritmo com as otimizações sugeridas. Pode-se também fazer a análise de pior caso da complexidade de espaço do algoritmo em [ND07] para $K_{(n)}$ e para as outras 15 famílias de lógica modal, comparando os resultados com outros algoritmos.

Referências

[Che80] Brian F. Chellas. *Modal Logic: an introduction*. Cambridge University Press, 1980. 6

[CLRS02] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2002. 3

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158. ACM, 1971. 1

[FHMV95] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995. 1

[GKWZ03] Dov Gabbay, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. *Many-dimensional modal logics: theory and applications*. Studies in Logic, 148. Elsevier Science, 2003. 1

[HC01] G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, 2001. 6

[LAD77] Richard E. LADNER. The computational complexity of provability in systems of modal propositional logic. *SIAM J. COMPUT*, 6(3), 1977. 1, 12, 23, 28

[NCW95] S. Nienhuys-Cheng and R. Wolf. The subsumption theorem in inductive logic programming: Facts and fallacies. In *Advances in Inductive Logic Programming. IOS*, pages 265–276. Press, 1995. 13

[ND06] Cláudia Nalon and Clare Dixon. Anti-prenexing and prenexing for modal logics. In *European Conference on Logics in Artificial Intelligence*, pages 333–345, 2006. 8, 9

[ND07] Cláudia Nalon and Clare Dixon. Clausal resolution for normal modal logics. *Journal of Algorithms*, 62:117–134, 2007. 1, 6, 10, 11, 12, 23, 28

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994. 3

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12:23–41, January 1965. 10

[Sip97] M. Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, 1997. 1, 5

[Tor04] Jacobo Torán. Space and width in propositional resolution (column: Computational complexity). *Bulletin of the EATCS*, 83:86–104, 2004. 1

[Tur36] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of The London Mathematical Society*, s2-42:230–265, 1936. 2