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

# Algoritmos de seleção de cláusulas em provas por resolução para lógicas modais

José Marcos da Silva Leite

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

Orientadora
Prof.a Dr.a Cláudia Nalon

Brasília
2021

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

# Algoritmos de seleção de cláusulas em provas por resolução para lógicas modais

José Marcos da Silva Leite

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

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

Prof. Dr. Guilherme Novaes Ramos
Universidade de Brasília

Prof. Dr. Vinicius Ruela Pereira Borges
Universidade de Brasília

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

Brasília, 11 de novembro de 2021

# Dedicatória

*Eu dedico essa música a primeira garota que tá sentada ali na fila. Brigado!*

# Agradecimentos

Nos *agradecimentos*

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

# Resumo

O *resumo*

**Palavras-chave:** LaTeX, metodologia científica, trabalho de conclusão de curso

# Abstract

O *abstract* é o resumo

**Keywords:** LaTeX, scientific method, thesis

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Em computação, estudamos classes de problemas que ajudam a expressar quão difícil resolver um problema é, classificando por tempo ou espaço. P é a classe de problemas que podem ser resolvidos em tempo polinomial em uma máquina determinística. NP é a classe de problemas que podem ser resolvidos em tempo polinomial em uma máquina não deterministica. O problema de P vs NP consiste em determinar se P $\neq$ NP e tem implicações na matemática, criptografia. PSPACE é a classe de problemas que podem ser resolvidos em espaço polinomial. Dentre outras, as classes P e NP estão contidas dentro da classe PSPACE o que demonstra um pouco a sua representatividade. Logica Modal proposicional $K_n$ pode expressar qualquer problema da classe PSPACE, ou seja, PSPACE-completo [**?**].

Representar problemas formalmente nos permite usar ferramentas para garantidamente formular ou reconhecer uma resposta. A grande expressividade de $K_n$ permite minimização de Automatos Finitos Não determinísticos[**?**], computar o Equilíbrio de Nash para qualquer jogo de dois jogadores em forma normal [**?**], ou até representar modelos de outras lógicas [3] o que possibilita seu uso desde engenharia de requisitos [1] a geração contos de fadas [2].

Assim, é desejável ter ferramentar para resolver problemas em $K_n$ e que estas sejam eficientes. Para isso, muitos provadores de teoremas foram construídos para tal lógica. KSP é um provador automático baseado em resolução descrito em [6]. Provadores automáticos são interessantes por diminuir o risco de erro humano na prova. Um dos passos de todo provador baseado em resolução é a seleção de cláusula. Muitas heurísticas para tal já foram estudadas mas ainda há muito a ser melhorado mesmo ao estado da arte [4].

Neste trabalho queremos resolver eficientemente o problema de seleção de cláusula no KSP.

No Capítulo 2 apresentamos toda a teoria necessária para o trabalho. O Capítulo 3 descreve os experimentos feitos.

# Capítulo 2

# Definições

Nesta seção apresentamos as definições básicas para o resto do texto.

## 2.1 Linguagem

Trabalharemos com a linguagem lógica modal $K_n$.

**Definição 1** Seja $P = \{p, q, r, \dots\}$ um conjunto enumerável de símbolos proposicionais, $\mathcal{A} = \{1, 2, 3, \dots, n\}, n \in \mathbb{N}$. Definimos o conjunto de fórmulas $\mathcal{FBF}$ indutivamente.

- Se $\varphi \in \mathcal{P}$ então $\varphi \in \mathcal{FBF}$
- Se $\varphi \in \mathcal{FBF}$, $\psi \in \mathcal{FBF}$ e $a \in \mathcal{A}$, então $(\varphi \wedge \psi) \in \mathcal{FBF}$, $(\varphi \vee \psi) \in \mathcal{FBF}$, $(\varphi \rightarrow \psi) \in \mathcal{FBF}$, $\boxed{a}\varphi \in \mathcal{FBF}$, $\diamondsuit_a\varphi \in \mathcal{FBF}$ e $\neg\varphi \in \mathcal{FBF}$

**Definição 2** Denotamos por $\mathcal{LP}$ o conjunto de literais proposicionais e por $\mathcal{LM}$ o conjunto de literais modais. $\forall p \in \mathcal{P}, \forall a \in \mathcal{A}$, temos que $p \in \mathcal{LP}, \neg p \in \mathcal{LP}, \boxed{a}p \in \mathcal{LM}, \boxed{a}\neg p \in \mathcal{LM}, \diamondsuit_a p \in \mathcal{LM}, \diamondsuit_a \neg p \in \mathcal{LM}$

**Definição 3** Uma cláusula proposicional é uma disjunção de literais proposicionais.

Assim, uma cláusula proposicional tem a forma $p_1 \vee p_2 \vee \cdots p_m$ com $p_i \in \mathcal{P}$.

Seja $\Sigma = \{0, 1\}$, $\Sigma^*$ é o conjunto de todas as cadeias formadas com elementos de $\Sigma$. Em particular, $\epsilon$ representa a cadeia vazia. Construiremos cadeias em $\Sigma^*$ para codificar a posi-

ção de ocorrência de uma subfórmula em uma fórmula. Seja $inv$: {positiva, negativa} $\mapsto$ {positiva, negativa} tal que $inv$(positiva) = negativa e $inv$(negativa) = positiva.

**Definição 4** Definimos a polaridade de uma subfórmula pela função $pol: \mathcal{FBF} \times \mathcal{FBF} \times \Sigma^* \mapsto$ {positiva, negativa}. Para $\varphi, \chi_1, \chi_2 \in \mathcal{FBF}, s \in \Sigma^*, a \in \mathcal{A}, val \in$ {positiva, negativa}.

- $pol(\varphi, \varphi, \epsilon)$ = positiva.
- Se $pol(\varphi, \chi_1 \vee \chi_2, s) = val$, então $pol(\varphi, \chi_1, s0) = pol(\varphi, \chi_2, s1) = val$
- Se $pol(\varphi, \chi_1 \wedge \chi_2, s) = val$, então $pol(\varphi, \chi_1, s0) = pol(\varphi, \chi_2, s1) = val$
- Se $pol(\varphi, \chi_1 \rightarrow \chi_2, s) = val$, então $pol(\varphi, \chi_1, s0) = inv(val)$ e $pol(\varphi, \chi_2, s1) = val$
- Se $pol(\varphi, \diamondsuit_a \chi_1, s) = val$, então $pol(\varphi, \chi_1, s0) = val$
- Se $pol(\varphi, \boxed{a}\chi_1, s) = val$, então $pol(\varphi, \chi_1, s0) = val$
- Se $pol(\varphi, \neg\chi_1, s) = val$, então $pol(\varphi, \chi_1, s0) = inv(val)$

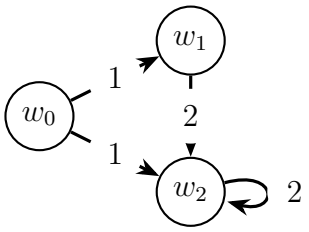
Dizemos que a polaridade de $\chi_1$ em $\varphi$ na posição $s$ é $pol(\varphi, \chi_1, s)$.

**Definição 5** Definimos o nível modal de uma subfórmula pela função $mlevel: \mathcal{FBF} \times \mathcal{FBF} \times \Sigma^* \mapsto \mathbb{N}$. Para $\varphi, \chi_1, \chi_2 \in \mathcal{FBF}, s \in \Sigma^*, a \in \mathcal{A}, val \in \mathbb{N}$.

- $mlevel(\varphi, \varphi, \epsilon) = 0$.
- Se $mlevel(\varphi, \chi_1 \vee \chi_2, s) = val$ ou $mlevel(\varphi, \chi_1 \wedge \chi_2, s) = val$ ou $mlevel(\varphi, \chi_1 \rightarrow \chi_2, s) = val$, então $mlevel(\varphi, \chi_1, s0) = mlevel(\varphi, \chi_2, s1) = val$
- Se $mlevel(\varphi, \diamondsuit_a \chi_1, s) = val$ ou $mlevel(\varphi, \boxed{a}\chi_1, s) = val$, então $mlevel(\varphi, \chi_1, s0) = val + 1$
- Se $mlevel(\varphi, \neg\chi_1, s) = val$, então $mlevel(\varphi, \chi_1, s0) = val$

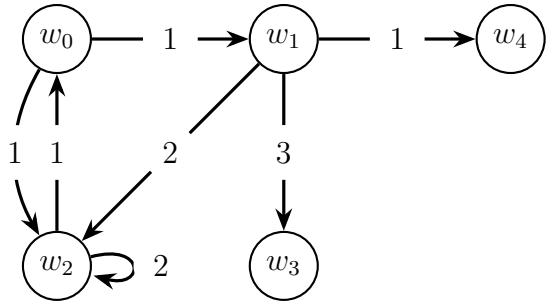
Dizemos que o nível modal de $\chi_1$ em $\varphi$ na posição $s$ é $mlevel(\varphi, \psi, s)$.

A semântica para lógica modal proposicional é dada por estruturas de Kripke. Uma estrutura de Kripke $M$ é da forma $M = (\mathcal{W}, w_0, \mathcal{R}_1, \ldots, \mathcal{R}_{|\mathcal{A}|}, \pi)$, onde $\mathcal{W}$ é um conjunto de mundos possíveis, $w_0 \in \mathcal{W}$, $\pi : \mathcal{W} \times \mathcal{P} \rightarrow \{true, false\}$, $\mathcal{R}_a \subseteq \mathcal{W} \times \mathcal{W}$ para todo $a \in \mathcal{A}$. Dizemos que uma fórmula $\varphi$ é satisfeita na lógica modal K no modelo $M$ no mundo $w$ se,




| $\pi$ | $w_0$ | $w_1$ | $w_2$ |
|---|---|---|---|
| p | *false* | *true* | *false* |
| q | *false* | *true* | *true* |

Figura 2.1: Primeiro exemplo de modelo.




| $\pi$ | $w_0$ | $w_1$ | $w_2$ | $w_3$ | $w_4$ |
|---|---|---|---|---|---|
| p | *false* | *false* | *true* | *true* | *false* |
| q | *false* | *false* | *false* | *false* | *false* |

Figura 2.2: Segundo exemplo de modelo.

e somente se, $\langle M, w\rangle \models \varphi$, conforme segue:

- $\langle M, w\rangle \models p$, se e somente se $\pi(w, p) = \mathit{true}$, para $p \in \mathcal{P}$
- $\langle M, w\rangle \models \neg\varphi$, se e somente se $\langle M, w\rangle \not\models \varphi$
- $\langle M, w\rangle \models (\varphi \wedge \psi)$, se e somente se $\langle M, w\rangle \models \varphi$ e $\langle M, w\rangle \models \psi$
- $\langle M, w\rangle \models (\varphi \vee \psi)$, se e somente se $\langle M, w\rangle \models \varphi$ ou $\langle M, w\rangle \models \psi$
- $\langle M, w\rangle \models (\varphi \rightarrow \psi)$, se e somente se $\langle M, w\rangle \not\models \varphi$ ou $\langle M, w\rangle \models \psi$
- $\langle M, w\rangle \models \diamondsuit_a\varphi$, se e somente se $\exists w', (w, w') \in \mathcal{R}_a, \langle M, w'\rangle \models \varphi$
- $\langle M, w\rangle \models \boxed{a}\varphi$, se e somente se $\forall w', (w, w') \in \mathcal{R}_a, \langle M, w'\rangle \models \varphi$

Uma fórmula $\varphi$ é localmente satisfatível se existe um modelo $M$ tal que $\langle M, w_0\rangle \models \varphi$. Uma formula $\varphi$ é globalmente satisfatível se existe um modelo $M$ tal que para todo $w \in \mathcal{W}$ temos que $\langle M, w\rangle \models \varphi$. Escrevemos $M \models \varphi$ se, e somente se, $\langle M, w_0\rangle \models \varphi$.

Na figura 2.1 apresentamos a visualização de um modelos possível para a fórmula $(p \wedge \boxed{1}(p \wedge \diamondsuit_2 q)$. Na figura 2.2, um modelo para $(\boxed{1}\boxed{2}\diamondsuit_1\diamondsuit_1\boxed{3}(p \rightarrow q)) \vee (\diamondsuit_1\boxed{3}(p \rightarrow q) \wedge \boxed{1}\boxed{1}(\diamondsuit_1\neg q \vee \boxed{1}q)$.

**Definição 6** Para um modelo $M$, definimos a profundidade de um mundo pela função $depth\colon \mathcal{W} \mapsto \mathbb{N}$, onde $depth(w)$ é tamanho do menor caminho de $w_0$ a $w$ no grafo $(\mathcal{W}, \bigcup_{i=1}^{n} R_i)$. Chamamos de nível modal a classe de equivalência com todos os mundos com mesma profundidade.

Podemos reduzir o problema de satisfatibilidade global ao problema de satisfatibilidade local com a extensão da linguagem K pelo operador universal $\boxed{*}$. Seja $M = (\mathcal{W}, w_0, \mathcal{R}_1, \ldots, \mathcal{R}_{|\mathcal{A}|}, \pi)$, $\langle M, w\rangle \models \boxed{*}\varphi$ se, e somente se, para todo $w' \in \mathcal{W}$, $\langle M, w'\rangle \models \varphi$.

**Definição 7** Uma fórmula está na forma normal negada caso seja formada somente por símbolos proposicionais, $\neg$, $\wedge$, $\vee$, $\boxed{a}$ e $\diamondsuit_{\varphi}$ para $a \in \mathcal{A}$, e a negação só é aplicada a símbolos proposicionais.

É importante ressaltar se $\varphi \in \mathcal{FBF}$ não está na forma normal negada, então $\varphi$ pode ser reescrita como $\psi \in \mathcal{FBF}$ na forma normal negada com semântica equivalente. Isto é, para todo $\langle M, w\rangle$, $\langle M, w\rangle \models \varphi$ se, e somente se, $\langle M, w\rangle \models \psi$.

## 2.2 Forma Normal em Camadas

O cálculo a ser apresentado utiliza uma outra linguagem chamada de Forma Normal Separada em Níveis Modais ($SNF_{ml}$).

**Definição 8** Uma fórmula em $SNF_{ml}$ é uma conjunção de cláusulas. Para $ml \in \mathbb{N} \cup \{*\}$ e $l_1, l_2 \in \mathcal{LP}$, cada cláusula está em um dos três formatos:

- $ml : c$, onde $c$ é uma cláusula proposicional
- $ml : l_1 \rightarrow \boxed{a} l_2$
- $ml : l_1 \rightarrow \diamondsuit_{\varphi} l_2$

A satisfatibilidade de uma fórmula em $SNF_{ml}$ é definida a partir da satisfatibilidade de $K_n$. Sejam $ml : \varphi$ e $ml : \psi$ cláusulas $SNF_{ml}$ e $M$ um modelo na lógica $K_n$.

- $M \models * : \varphi$ se, e somente se, $M \models \boxed{*}\varphi$.
- $M \models (ml : \varphi) \wedge (ml : \psi)$ se, e somente se, $M \models ml : \varphi$ e $M \models ml : \psi$.
- $M \models ml : \varphi$ se, e somente se, para todo $w$ tal que $depth(w) = ml$, temos $\langle M, w\rangle \models ml : \varphi$.

Uma função de tradução de $K_n$ para $SNF_{ml}$ bem como prova de que a tradução de uma fórmula preserva satisfatibilidade podem ser encontradas em [5].

Seja $\geq$ uma ordem total sobre os símbolos proposicionais. Estendemos esta ordem para os literais da seguinte forma: Se $p \in \mathcal{P}$, então $\neg p \geq p$; Se $p, q \in \mathcal{P}, p \neq q$ e $p \geq q$, então $\neg p \geq q$.

**Definição 9** O literal $l$ é máximo em $\varphi \in SNF_{ml}$ se e somente se $l$ ocorre em $\varphi$ e não há $l_2 \neq l$ em $\varphi$ tal que $l_2 \geq l$.

Note que podemos escolher qualquer ordem sobre os símbolos proposicionais, assim $l$ pode ser máximo numa ordem e não ser máximo em outra ordem.

**Definição 10** O tamanho de uma cláusula em $SNF_{ml}$ é a cardinalidade do conjunto contendo somente os literais na cláusula.

## 2.3 Regras de inferência

O cálculo dedutivo baseado em resolução $RES_{ml}$ para lógica $K_n$ foi descrito em [5]. Para simplificar a descrição das regras de inferência faremos uso de uma função parcial de unificação $\sigma\colon P(\mathbb{N} \cup \{*\}) \mapsto \mathbb{N} \cup \{*\}$, tal que $\sigma(\{ml, *\}) = ml$, $\sigma(\{ml\}) = ml$, e indefinida caso contrário. As regras de inferência de $RES_{ml}$ são apresentadas na Figura 2.3, onde $* - 1 = *$ e $m$ pode ser 0. Essas regras só valem se o resultado da unificação for definido. Demostrações de correção e corretude podem ser encotradas em [5].

## 2.4 Algoritmo

Neste trabalho usaremos o KSP [6], um provador baseado no cálculo visto na Seção 2.3 e determina a satisfatibilidade de fórmulas em $K_n$. Caso insatisfatível, uma prova é fornecida. Embora o KSP tome fórmulas em $K_n$ como entrada, este faz a tradução para $SNF_{ml}$ e todo o processamento é feito nessa linguagem.

KSP utiliza uma variação de conjunto de suporte, técnica que restringe os candidatos possíveis para resolução. A demostração da correção e completude dessa extensão para $SNF_{ml}$ pode ser encontrada em [5].

Na versão para lógica clássica de conjunto de suporte, o conjunto de cláusulas $\Delta$ é particionado em dois conjuntos $\Gamma$, o conjunto de suporte ou não processado, e $\Lambda$, o conjunto *usable* ou processado. Cláusulas são selecionadas de $\Gamma$ e resolvidas com cláusulas em $\Lambda$.

[LRES]
$$\frac{ml : (D \vee l) \quad ml' : (D' \vee \neg l)}{\sigma(\{ml, ml'\}) : D \vee D'}$$

[MRES]
$$\frac{ml : (l_1 \rightarrow \boxed{a} l) \quad ml' : (l_2 \rightarrow \diamondsuit_a \neg l)}{\sigma(\{ml, ml'\}) : \neg l_1 \vee \neg l_2}$$

[GEN2]
$$\frac{ml_1 : (l'_1 \rightarrow \boxed{a} l_1) \quad ml_2 : (l'_2 \rightarrow \boxed{a} \neg l_1) \quad ml_3 : (l'_3 \rightarrow \diamondsuit_a l_2)}{\sigma(\{ml_1, ml_2, ml_3\}) : \neg l'_1 \vee \neg l'_2 \vee \neg l'_3}$$

[GEN1]
$$\frac{\begin{array}{c} ml_1 : (l'_1 \rightarrow \boxed{a} l_1) \\ \vdots \\ ml_m : (l'_m \rightarrow \boxed{a} \neg l_m) \\ ml_{m+1} : (l' \rightarrow \diamondsuit_a \neg l) \\ ml_{m+2} : (l_1 \vee \ldots \vee l_m \vee l) \end{array}}{\begin{array}{c} ml : \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l' \\ ml = \sigma(\{ml_1, \ldots, ml_{m+1}, ml_{m+2} - 1\}) \end{array}}$$

[GEN3]
$$\frac{\begin{array}{c} ml_1 : (l'_1 \rightarrow \boxed{a} l_1) \\ \vdots \\ ml_m : (l'_m \rightarrow \boxed{a} \neg l_m) \\ ml_{m+1} : (l' \rightarrow \diamondsuit_a l) \\ ml_{m+2} : (l_1 \vee \ldots \vee l_m) \end{array}}{\begin{array}{c} ml : \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l' \\ ml = \sigma(\{ml_1, \ldots, ml_{m+1}, ml_{m+2} - 1\}) \end{array}}$$

Figura 2.3: Regras de inferência

A cláusula selecionada é removida de $\Gamma$ e acrescentada a $\Lambda$. Os resolventes produzidos são inseridos em $\Gamma$.

Na extensão para $SNF_{ml}$, onde as cláusulas são rotuladas pelo nível modal, as cláusulas de todo nível modal $ml$ são particionadas em três conjuntos $\Gamma^{lit}_{ml}$, $\Lambda^{lit}_{ml}$ e $\Lambda^{mod}_{ml}$. Cláusulas proposicionais são particionadas em $\Gamma^{lit}_{ml}$ e $\Lambda^{lit}_{ml}$ como no caso em lógica clássica. Cláusulas modais são armazenadas em $\Lambda^{mod}_{ml}$. Note que nenhuma regra de inferência descrita na Seção 2.3 produz novas cláusulas modais.

A seguir, descrevemos o algoritmo implementado no KSP.

**Algoritmo 1:** KSP-Proof-Search

**Entrada:** Fórmula em $K_n$

**Saída:** Satisfatibilidade da fórmula

```
1  preprocessamento-da-entrada;
2  tradução-para-SNF;
3  preprocessamento-de-clausulas;
4  Γ^lit ← ⋃Γ^lit_ml;
5  enquanto Γ^lit ≠ ∅ faça
6      para todo nível modal ml faça
7          clausula ← given(ml);
8          se não redundante(clausula) então
9              GEN1(clausula, ml, ml − 1);
10             GEN3(clausula, ml, ml − 1);
11             LRES(clausula, ml, ml);
12             Λ^lit_ml ← Λ^lit_ml ∪ {clausula};
13         fim
14         Γ^lit_ml ← Γ^lit_ml \ {clausula};
15         se 0 : false ∈ Γ^lit_0 então
16             retorna insatisfatível;
17         fim
18         Γ^lit ← ⋃Γ^lit_ml;
19     fim
20     retorna satisfatível;
21 fim
```

As Linhas 1-3 aplicam algumas regras de simplificação, traduzem a fórmula para linguagem $SNF_{ml}$ e constroem os conjuntos *usable* e de suporte. As Linhas 9-11 aplicam regras de inferências descritas na Seção 2.3.

A função given, Linha 7, é responsável por escolher uma cláusula dentre todas as candidatas possíveis do conjunto de suporte. Cada nível modal é independente. Naturalmente, a função given só considera as cláusulas do nível modal pedido. KSP implementa cinco variações dessa função: *menor*, *mais antiga*, *mais nova*, *mínima* e *máxima*; e o usuário pode escolher qual deseja utilizar.

Na variação *menor*, é selecionada uma cláusula com o menor tamanho de $\Gamma^{lit}_{ml}$.

Na variação *mais antiga*, é salvo a ordem nas quais as cláusulas foram adicionadas ao conjunto de cláusulas e é selecionada a que foi adicionada antes de todas em $\Gamma^{lit}_{ml}$.

*Mais nova* é análoga a *mais antiga*, mas é selecionada a que foi adicionada depois de todas as outras.

Em *mínima*, é escolhida uma cláusula com o menor tamanho dentre as cláusulas com o menor literal máximo em $\Gamma_{ml}^{lit}$.

Em *máxima*, é feita escolha análoga a *mínima* mas dentre cláusulas com o maior literal máximo em $\Gamma_{ml}^{lit}$.

Neste trabalho propomos novos métodos para seleção de cláusulas e comparamos com os métodos previamente utilizados no KSP. Para comparação usaremos o LWB [7], que descreve geradores de *benchmark* de tamanho arbitrário para 9 famílias de fórmulas.

# Capítulo 3

# Proposta de Solução

Neste capítulo propomos novos métodos de seleção de cláusula para o KSP.

Fizemos um experimento inicial para averiguar o desempenho do KSP sobre o LWB com os algoritmos de seleção de cláusula já implementados. O sistema utilizado tem processador AMD FX-6300 com clock base de 3.5 GHz, 6 Gigabyte de memória RAM no Sistema Operacional Ubuntu 18.04. Escolhemos a versão 0.1.2 do KSP, a versão pública mais recente na data do experimento. Nesse experimento usamos 21 fórmulas satisfatíveis e 21 insatisfatíveis para cada família. Para cada fórmula foram dados 10 segundos de tempo limite.

A Tabela 3.1 mostra o resultado desse experimento. Para cada família de fórmulas temos uma linha na tabela para as satisfatíveis, representada pelo sufixo _n, e uma linha para as insatisfatíveis, representada pelo sufixo _p. Cada coluna representa uma heurística de seleção de cláusula. Cada célula da tabela informa a quantidade de fórmulas que o provador proveu solução dentro do tempo limite e a média de tempo entre essas.

Esses resultados indicam que o KSP é muito eficiente para a maioria das famílias independente da estratégia utilizada, mas muito pode ser melhorado em k_branch, k_d4, k_ph e k_poly.

## 3.1 Trabalhos anteriores

Uma análise de heurísticas de seleção de cláusula para provadores baseados em saturação, assim como KSP, foi feita por [4]. O experimento foi feito no provador E sobre 13774 fórmulas do *benchmark* TPTP [8] com 300 segundos de tempo limite.

As heurísticas avaliadas foram: *Mais antiga*, Contagem de Símbolos e Ordenada. Em Contagem de Símbolos, é atribuído um peso a cada símbolo e é selecionada uma cláusula com a menor soma de pesos. No caso em que todos os símbolos têm peso um esta variação

| – | *mais antiga* | *mais nova* | *mínima* | *máxima* | *menor* |
|---|---|---|---|---|---|
| k_branch_n | 2(59.6) | 1(0.06) | 1(0.00) | 1(0.05) | 2(4.34) |
| k_branch_p | 2(1.52) | 2(88.84) | 2(29.37) | 2(101.32) | 3(1.4) |
| k_d4_n | 9(14.5) | 7(44.42) | 8(27.48) | 8(21.68) | 9(13.5) |
| k_d4_p | 21(0.16) | 21(0.08) | 21(0.03) | 21(0.57) | 21(0.08) |
| k_dum_n | 21(0.03) | 21(0.05) | 21(0.04) | 21(0.12) | 21(0.08) |
| k_dum_p | 21(0.09) | 21(0.14) | 21(0.51) | 21(0.34) | 21(0.06) |
| k_grz_n | 21(9.56) | 17(3.01) | 17(5.9) | 17(16.35) | 21(3.15) |
| k_grz_p | 21(0.02) | 21(0.01) | 21(0.01) | 21(0.01) | 21(0.01) |
| k_lin_n | 21(0.01) | 21(0.01) | 21(0.01) | 21(0.01) | 21(0.02) |
| k_lin_p | 21(0.01) | 21(0.01) | 21(0.01) | 21(0.01) | 21(0.03) |
| k_path_n | 21(0.02) | 21(0.02) | 21(0.02) | 21(0.02) | 21(0.03) |
| k_path_p | 21(0.03) | 21(0.02) | 21(0.02) | 21(0.02) | 21(0.02) |
| k_ph_n | 3(4.08) | 3(10.88) | 3(1.85) | 3(5.81) | 3(0.65) |
| k_ph_p | 2(0.02) | 2(0.00) | 3(14.82) | 3(89.8) | 3(0.03) |
| k_poly_n | 16(50.45) | 8(27.89) | 14(42.53) | 13(39.82) | 16(52.46) |
| k_poly_p | 15(36.83) | 10(38.17) | 14(45.65) | 14(39.16) | 15(43.32) |
| k_t4p_n | 21(0.06) | 21(0.16) | 21(0.1) | 21(0.06) | 21(0.05) |
| k_t4p_p | 21(0.04) | 21(0.1) | 21(0.03) | 21(0.04) | 21(0.02) |

Tabela 3.1: Fórmulas resolvidas em até 300 seg e tempo médio em segundos.

é idêntica a *menor* usada no KSP. Ordenada é uma variação de Contagem de Símbolos onde é preferida cláusulas com o menor número de literais maximais.

Também foram analisadas várias intercalações de duas dessas heurísticas em distribuições diferentes. Por exemplo, a cada 11 seleções de cláusulas, 10 são feitas pela heurística Contagem de Símbolo e 1 é feita pela heurística *Mais antiga*.

São apresentados vários resultados como número de fórmulas resolvidas, tamanho da prova, número de inferências na prova, etc para todas as estratégias utilizadas. Vemos que a maioria das fórmulas resolvidas foram resolvidas em poucos segundos mesmo com 300 segundos disponíveis. Os experimentos apontam não haver melhora em performance ao usar diferentes funções de peso para os literais. Todas as estratégias de contagem de símbolos tiveram ganho significativo de desempenho quando intercaladas com *Mais antiga*. Selecionar sempre cláusulas dadas na entrada primeiro melhorou performance no geral, mas não tanto com estratégias utilizando *Mais antiga*.

## 3.2 Intercalação estática de estratégias

Baseado no trabalho de [4], nossa primeira proposta de seleção de cláusula será intercalação de *menor* e *mais antiga*, já implementadas no KSP, numa razão $p$:$q$ informada pelo usuário. Dessa forma, as primeiras $p$ execuções da função given serão conforme *menor*,

| – | Tempo médio(sat) | Resolvidas(sat) | Tempo médio(unsat) | Resolvidas(unsat) |
|---|---|---|---|---|
| k_branch | 5.23 | 2 | 3.68 | 3 |
| k_d4 | 13.93 | 9 | 0.021 | 21 |
| k_ph | 1.46 | 3 | 0.05 | 3 |
| k_poly | 60.24 | 16 | 38.16 | 15 |

Tabela 3.2: Fórmulas resolvidas com razão 5:1 entre tamanho e mais antiga em até 300 segundos e tempo médio em segundos.

as próximas $q$ serão conforme *mais antiga*, as próximas $p$ conforme *menor* e assim por diante.

A implementação dessa proposta não é muito complexa por ser intercalação de métodos já implementados e nos permite comparar os resultados do KSP com os encontrados por [4] no provador E.

Na tabela 3.2, apresentamos os resultados para razão de 5 seleções por *Menor tamanho* para cada seleção por *Mais antiga*. Usamos sat como abreviação de satisfatível e unsat como abreviação de unsatisfatível.

## 3.3 Intercalação dinâmica de estratégias

Como a seleção de cláusula em cada nível é feita de forma independente, podemos também usar algoritmos distintos em níveis distintos.

Com base nos resultados dos experimentos descritos na Seção 3.2, propomos intercalação de *menor* e *mais antiga* com razão dinâmica para todo nível.

Cada nível terá um escalonador responsável por escolher uma razão eficiente. Como não conhecemos a melhor solução a priori, esta razão mudará ao longo da execução do provador. Neste experimento usamos somente o número de inferências feitas para determinar a troca de proporção na intercalação dos algoritmos.

Para o nível modal $ml$, o escalonador usará apenas a cardinalidade de $\Gamma^{lit}_{ml}$ para determinar qual razão aplicar. O usuário define $r_1, l_1, r_2, l_2, \ldots, r_x$, onde $r_i$ é uma razão descrita na Seção 3.2 e $l_i$ é uma sequência ordenada de limiares para troca de razões. Para cardinalidade menor ou igual a $l_1$ é usada razão $r_1$, para cardinalidade maior que $l_1$ e menor ou igual a $l_2$ é usada razão $r_2$ e assim por diante. Note que não há limite superior para a ultima razão, a $r_x$.

| – | Tempo médio(sat) | Resolvidas(sat) | Tempo médio(unsat) | Resolvidas(unsat) |
|---|---|---|---|---|
| k_branch | 5.27 | 2 | 11.64 | 3 |
| k_d4 | 14.89 | 9 | 0.03 | 21 |
| k_ph | 2.04 | 3 | 0.02 | 3 |
| k_poly | 53.04 | 16 | 37.76 | 15 |

Tabela 3.3: Fórmulas resolvidas em até 300 segundos e tempo médio em segundos com intecalação dinâmica.

| – | Tempo médio(sat) | Resolvidas(sat) | Tempo médio(unsat) | Resolvidas(unsat) |
|---|---|---|---|---|
| k_branch | 3.17 | 2 | 0.23 | 3 |
| k_d4 | 16.94 | 9 | 0.03 | 21 |
| k_ph | 1.25 | 3 | 0.02 | 3 |
| k_poly | 40.03 | 15 | 37.34 | 15 |

Tabela 3.4: Fórmulas resolvidas em até 300 segundos e tempo médio em segundos com intercalação dinâmica agregada.

## 3.4 Intercalação dinâmica agregada à preferência literais

A regra de inferência GEN1 no laço principal do KSP precisa de alguma cláusula com um literal que aparece num operador $\diamondsuit_\alpha$, para qualquer $\alpha$. Para facilitar esta aplicação, vamos fazer seleção entre cláusulas com algum literal destes caso exista. Por exemplo, para o algoritmo de *Menor tamanho*, vamos selecionar a cláusula de menor tamanho dentre as que têm algum literal de $\diamondsuit_\alpha$.

# Capítulo 4

# Conclusão

# Referências


[1] Castaneda, Veronica, Luciana Ballejos, Ma. Laura Caliusco e Ma. Rosa Galli: *The use of ontologies in requirements engineering*. Global Journal of Research In Engineering, 10(6), 2010, ISSN 2249-4596. https://www.engineeringresearch.org/index.php/GJRE/article/view/76. 1

[2] Peinado, Federico e Belen iaz Agudo: *A description logic ontology for fairy tale generation*. janeiro 2004. 1

[3] Schild, Klaus: *A correspondence theory for terminological logics: Preliminary report*. Em *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'91, página 466–471, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc., ISBN 1558601600. 1

[4] Schulz, Stephan e Martin Möhrmann: *Performance of clause selection heuristics for saturation-based theorem proving*. Em Olivetti, Nicola e Ashish Tiwari (editores): *Automated Reasoning*, páginas 330–345, Cham, 2016. Springer International Publishing, ISBN 978-3-319-40229-1. 1, 10, 11, 12

[5] Nalon, Cláudia, Clare Dixon e Ullrich Hustadt: *Modal resolution: Proofs, layers, and refinements*. ACM Transactions on Computational Logic, 20(4), agosto 2019, ISSN 1529-3785. https://doi.org/10.1145/3331448. 6

[6] Nalon, Cláudia, Ullrich Hustadt e Clare Dixon: *$K_S P$ a resolution-based theorem prover for $\mathsf{K}_n$: Architecture, refinements, strategies and experiments*. Journal of Automated Reasoning, 64(3):461–484, Mar 2020, ISSN 1573-0670. https://doi.org/10.1007/s10817-018-09503-x. 1, 6

[7] Balsiger, Peter, Alain Heuerding e Stefan Schwendimann: *A benchmark method for the propositional modal logics k, kt, s4*. J. Autom. Reason., 24(3):297–317, abril 2000, ISSN 0168-7433. https://doi.org/10.1023/A:1006249507577. 9

[8] Sutcliffe, Geoff: *The tptp problem library and associated infrastructure: ttthe fof and cnf parts, v3.5.0*. Journal of Automated Reasoning, 43(4):337–362, janeiro 2009, ISSN 0168-7433. 10