

Um Proveedor de Teoremas Multi-Estratégia

Adolfo Gustavo Serra Seca Neto e Marcelo Finger

Instituto de Matemática e Estatística - USP

Motivação

- Problema aberto em computação: $\mathcal{P} = \mathcal{NP}$?
- SAT (Satisfatibilidade de fórmulas em lógica clássica proposicional): primeiro problema \mathcal{NP} -completo (Cook, 1971)
- Um algoritmo polinomial não-determinístico pode ser transformado num algoritmo determinístico exponencial

Motivação

- Regras de inferência de sistemas de prova são não-determinísticas
- Provadores de teoremas levam tempo exponencial para provar problemas difíceis

Provedores Automáticos de Teoremas

- Em geral, implementam um sistema de prova para alguma lógica
 - Entrada: descrição de um problema (fórmulas em algum sistema lógico)
 - Saída: Sim ou não (+ prova no sistema lógico utilizado)

Provedores Automáticos de Teoremas

- A maioria dos PAT's implementa o método da resolução para a lógica clássica de primeira ordem (Otter, Vampire)
- Competições são organizadas para comparar os desempenhos dos PAT's

Entrada para PAT's

- Descrição do problema (fórmulas em algum sistema lógico)
- Exemplos:
 - $A, (A \rightarrow B) \vdash B$
 - $\top A \quad \top A \rightarrow B \quad \text{F } B$
 - $(A \wedge (A \rightarrow B)) \rightarrow (B \wedge C)$
 - $\circ A, A, \neg A \vdash (B \wedge C)$

Exemplo de entrada: PHP₂

- Fórmula:

$$\begin{aligned} & \mathbb{F} \left(\left((p_{0,0} \vee p_{0,1}) \wedge (p_{1,0} \vee p_{1,1}) \wedge (p_{2,0} \vee p_{2,1}) \right) \rightarrow \right. \\ & \left. \left((p_{0,0} \wedge p_{1,0}) \vee (p_{0,0} \wedge p_{2,0}) \vee (p_{1,0} \wedge p_{2,0}) \vee (p_{0,1} \wedge \right. \right. \\ & \left. \left. p_{1,1}) \vee (p_{0,1} \wedge p_{2,1}) \vee (p_{1,1} \wedge p_{2,1}) \right) \right) \end{aligned}$$

- Variáveis proposicionais: 6
- Fórmulas: 18
- Fórmulas assinaladas: 1
- Complexidade: 35

Tamanho da entrada

- Tamanho da cadeia de caracteres que representa o problema (menos significativo)
- Complexidade da lista de fórmulas - soma das complexidades de todas as fórmulas da lista
- Quantidade de
 - Variáveis proposicionais
 - Fórmulas e fórmulas assinaladas
 - Complexidade (ou quantidade de símbolos, *symbol-length*)

Complexidade de fórmulas

- $c(X A) = c(A)$ onde X é \top ou \perp
- $c(\neg A) = c(A) + 1$
- $c(A \circ B) = c(A) + c(B) + 1$, para qualquer \circ
- $c(A) = 1$ se A for uma variável proposicional

Lógicas e sistemas de prova

- Lógica clássica proposicional
 - Sistemas de prova corretos e completos: tabela-verdade, resolução, tablôs analíticos, tablôs KE, dedução natural, cálculo de seqüentes
- Lógica clássica de primeira ordem
 - Indecidível (tablôs podem não fechar mesmo com tautologia, laço infinito)
- Lógicas não-clássicas (modais, de inconsistência formal, etc.)

Método de tablôs

- Método refutacional - Para provar que A é válida, tenta-se provar que não é. Se chegamos somente a contradições, então A é válida.
- Árvore de prova é construída. Ramos fecham quando se encontra uma contradição. Uma árvore é dita fechada quando todos os seus ramos estão fechados
- Tablôs analisam as fórmulas sintaticamente

Método de tablôs

- Tablôs analíticos não simulam polinomialmente tabelas-verdade (que são exponenciais)
- A complexidade de provas baseadas em tablôs depende essencialmente do comprimento (número total de símbolos) da fórmula a ser decidida
- A complexidade das tabelas-verdade dependem apenas da quantidade de variáveis proposicionais distintas

Tabela-Verdade

A	B	C	$C \rightarrow A$	$B \rightarrow (C \rightarrow A)$	$(A \rightarrow (B \rightarrow (C \rightarrow A)))$
V	V	V	V	V	V
V	V	F	V	V	V
V	F	V	V	V	V
V	F	F	V	V	V
F	V	V	F	F	V
F	V	F	V	V	V
F	F	V	F	V	V
F	F	F	F	V	V

Tablôs Analíticos - prova

$\text{F}(A \rightarrow (B \rightarrow (C \rightarrow A)))$

$\text{T}A$

$\text{F}B \rightarrow (C \rightarrow A)$

$\text{T}B$

$\text{F}C \rightarrow A$

$\text{T}C$

$\text{F}A$

X

PAT's Baseados em Tablôs

- Saídas:
 1. Conseguiu provar (fechar o tablô)? Sim ou não.
 2. Prova (Objeto-prova ou árvore de prova):
 - Fórmulas e sua Origem (regra que originou e premissas utilizadas)

Complexidade de provas

Três principais noções de complexidade para provas (Buss):

- Número de linhas ou passos (em sistemas axiomáticos)
- Complexidade/comprimento/tamanho: quantidade de símbolos
- Profundidade: máxima profundidade E/OU de alguma fórmula que ocorre na prova

Complexidade de provas

Problema aberto (Buss): As tautologias têm provas de tamanho polinomial em sistemas axiomáticos? Isto é, existe um polinômio p tal que para todo $\psi \in TAUT$, existe uma prova Π de ψ de comprimento máximo $p(|\psi|)$? Se existir, então $\mathcal{P} = co\mathcal{NP}$!

Teoria da Complexidade de provas

- Está preocupada em provar limites inferiores não-triviais no comprimento das provas de classes de tautologias em sistemas de prova proposicionais corretos e completos.
- Provar limites inferiores superpolinomiais é bastante relevante para o estudo de ATP's

Teoria da Complexidade de provas

Se prova existe:

- Complexidade da menor prova possível
- Complexidade da busca pela prova

Um sistema de prova proposicional é automatizável se e somente se existe um procedimento determinístico para encontrar provas neste sistema em tempo polinomial, com relação à menor prova naquele sistema.

PAT's Baseados em Tablôs

- Medidas:
 - Tempo gasto pelo provador para encontrar a prova
 - Tamanho da prova:
 - Complexidade da árvore de prova: soma das complexidades de todas as *ocorrências* de fórmulas assinaladas nas árvores de prova
 - Altura da árvore de prova (considerando ramos como nós)

PAT's Baseados em Tablôs

- Medidas:
 - Tamanho da prova:
 - Comprimento do caminho mais longo da raiz até uma folha
 - Quantidade de fórmulas distintas
 - Quantidade de fórmulas assinaladas distintas

O Sistema KE

- Método de tablôs desenvolvido por Marco Mondadori e Marcello D'Agostino
- Baseado nos Tablôs Analíticos de Smullyan
- Possui regra PB baseada no Princípio da Bivalência, que é uma versão da regra do Corte do Cálculo de Seqüentes:
$$\frac{}{T A \mid F A}$$
- A regra PB não é eliminável
- Graças à regra PB, o sistema KE é mais eficiente que os TA's e permite uma diversidade maior de estratégias

Sistema KE - regras

$\frac{TA \vee B}{FA}$ TB	$FA \wedge B$ $\frac{TA}{FB}$	$TA \rightarrow B$ $\frac{TA}{TB}$	$\frac{T\neg A}{FA}$
$\frac{TA \vee B}{FB}$ TA	$\frac{TA \wedge B}{TA}$ TB	$TA \rightarrow B$ $\frac{FB}{FA}$	$\frac{F\neg A}{TA}$
$\frac{FA \vee B}{FA}$ FB	$FA \wedge B$ $\frac{TB}{FA}$	$FA \rightarrow B$ $\frac{TA}{FB}$	$\overline{TA FA}$

Prova mínima em KE

$\text{F } (A \rightarrow B1 \rightarrow B2 \rightarrow B3) | (A \rightarrow C1 \rightarrow C2 \rightarrow C3 \rightarrow A)$

$\text{F } (A \rightarrow B1 \rightarrow B2 \rightarrow B3)$

$\text{F } (A \rightarrow C1 \rightarrow C2 \rightarrow C3 \rightarrow A)$

$\text{T } A$

$\text{F } C1 \rightarrow C2 \rightarrow C3 \rightarrow A$

$\text{T } C1$

$\text{F } C2 \rightarrow C3 \rightarrow A$

\vdots

Prova mínima em KE

⋮

$T C2$

$F C3 \rightarrow A$

$T C3$

$F A$

x

Prova com passos desnecessários

$\text{F } (A \rightarrow B1 \rightarrow B2 \rightarrow B3) | (A \rightarrow C1 \rightarrow C2 \rightarrow C3 \rightarrow A)$

$\text{F } (A \rightarrow B1 \rightarrow B2 \rightarrow B3)$

$\text{F } (A \rightarrow C1 \rightarrow C2 \rightarrow C3 \rightarrow A)$

$\text{T } A$

$\text{F } C1 \rightarrow C2 \rightarrow C3 \rightarrow A$

$\text{F } B1 \rightarrow B2 \rightarrow B3(*)$

$\text{T } C1$

$\text{F } C2 \rightarrow C3 \rightarrow A$

\vdots

Prova com passos desnecessários

⋮

$\text{T } B1(*)$

$\text{F } B2 \rightarrow B3(*)$

$\text{T } C2$

$\text{F } C3 \rightarrow A$

$\text{T } B2(*)$

$\text{F } B3(*)$

$\text{T } C3$

$\text{F } A$

X

Exemplo de uso da regra PB

$T(a1 \vee b1)$

$T(a1 \rightarrow (a2 \vee b2))$

$T(b1 \rightarrow (a2 \vee b2))$

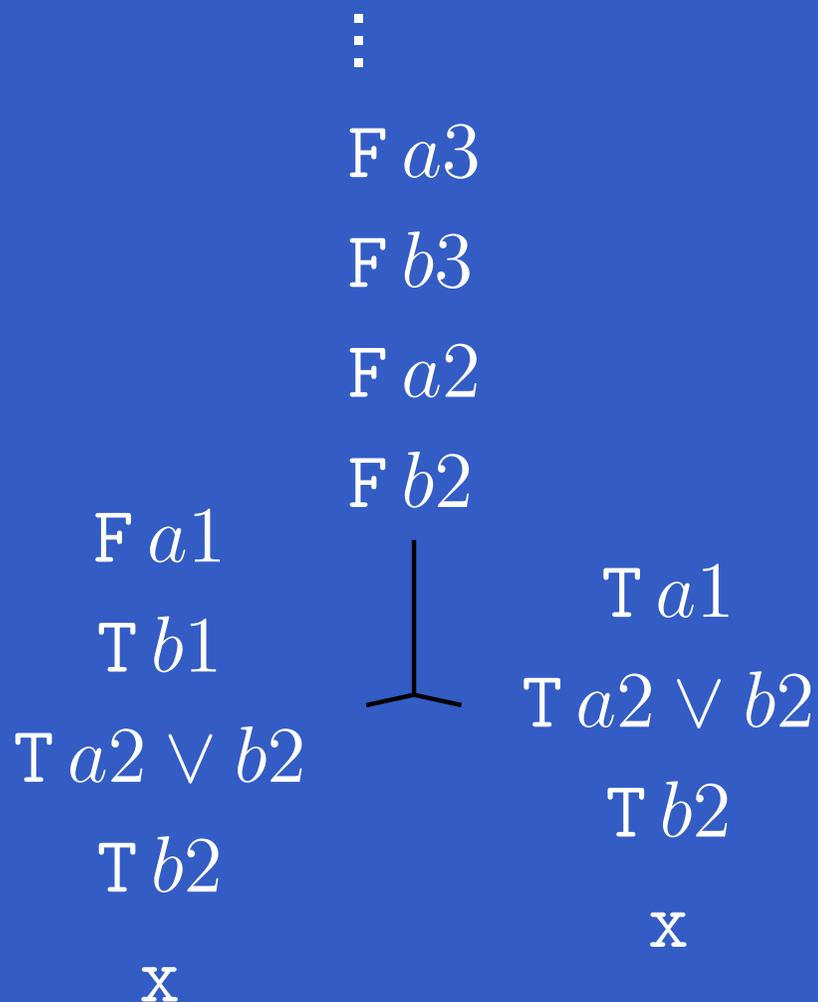
$T(a2 \rightarrow (a3 \vee b3))$

$T(b2 \rightarrow (a3 \vee b3))$

$F(a3 \vee b3)$

\vdots

Exemplo de uso da regra PB



Estratégias

- O que são estratégias?
- Ordem de aplicação das regras
- Algoritmos + Estruturas de dados
- Fechamento
- Representação de fórmulas
- Escolha de regras a partir de análise das fórmulas vs. tentativa-e-erro
- Diferentes conjuntos de regras

Regras simplificadoras

$$X \Phi(A \vee B)$$
$$F A$$
$$\frac{F A}{X \Phi(B)}$$
$$(X \vee F)$$
$$X \Phi(A \vee B)$$
$$T A$$
$$\frac{T A}{X \Phi(\top)}$$
$$(X \vee T)$$
$$X \Phi(\top \vee A)$$
$$\frac{X \Phi(\top \vee A)}{X \Phi(\top)}$$
$$(X \vee \top)$$
$$X \Phi(\perp \vee A)$$
$$\frac{X \Phi(\perp \vee A)}{X \Phi(A)}$$
$$(X \vee \perp)$$

Provedor de teoremas multi-estratégia

- Objetivo: Variar a estratégia utilizada usando a mesma implementação
- Aplicações:
 - Educacional: mostrar impacto da escolha da estratégia nas provas obtidas e no desempenho do sistema
 - Exploratória: desenvolver novas estratégias e compará-las
 - Adaptativa: provedor que escolhe estratégia de acordo com características do problema

O provedor multi-estratégia (MSTP)

- Versão atual: Lógica clássica proposicional
- KE + regras simplificadoras
- Orientação a Aspectos: Java e AspectJ

Primeira estratégia implementada

Enquanto não fechar a árvore de prova:

- Aplica todas as regras de uma premissa
- Aplica todas as regras de uma premissa simplificadoras
- Aplica todas as regras de duas premissas simplificadoras

Primeira estratégia implementada

Enquanto não fechar a árvore de prova:

- Aplica PB baseado em alguma fórmula não utilizada como premissa principal de uma regra de duas premissas. Põe o ramo direito na pilha de ramos abertos e tenta fechar o esquerdo. Volta ao início

Se fechar um ramo, pega o próximo na pilha.

Regras de uma premissa

- Fácil. Percorrer a lista de fórmulas assinaladas do ramo atual e, para cada fa, olhar o sinal e o conectivo. Verificar num mapeamento se há regra para esse par. Se houver aplica a regra e remove a fórmula da lista de fórmulas que podem ser usadas como premissa principal (PBCandidates)

Regras de uma premissa simplificadora

- Menos fácil. É preciso verificar se \top ou \perp é subfórmula de alguma fórmula e, nestas fórmulas, encontrar a 'superfórmula imediata'. Pegar o conectivo desta 'superfórmula imediata' e o símbolo (\top ou \perp) e consultar um mapeamento para verificar a regra que se aplica. Aplicar a regra e remove a fórmula de PBCandidates

Regras de duas premissas simplificador

- Para cada fórmula assinalada no ramo, verifica se entre as superfórmulas de sua fórmula existe alguma que faz parte de PBCandidates. Se houver, aplica regra e remove a fórmula de PBCandidates. Para aplicar a regra é preciso conhecer o conectivo da superfórmula e o sinal da fórmula assinalada.

Regra PB

- Pega a primeira fórmula assinalada de PBCandidates e aplica PB com a fórmula esquerda da regra de duas premissas simples que se aplicar.
- Caso não haja nenhuma fórmula assinalada em PBCandidates, o procedimento termina e o tabló permanece aberto.

Estruturas de dados utilizadas

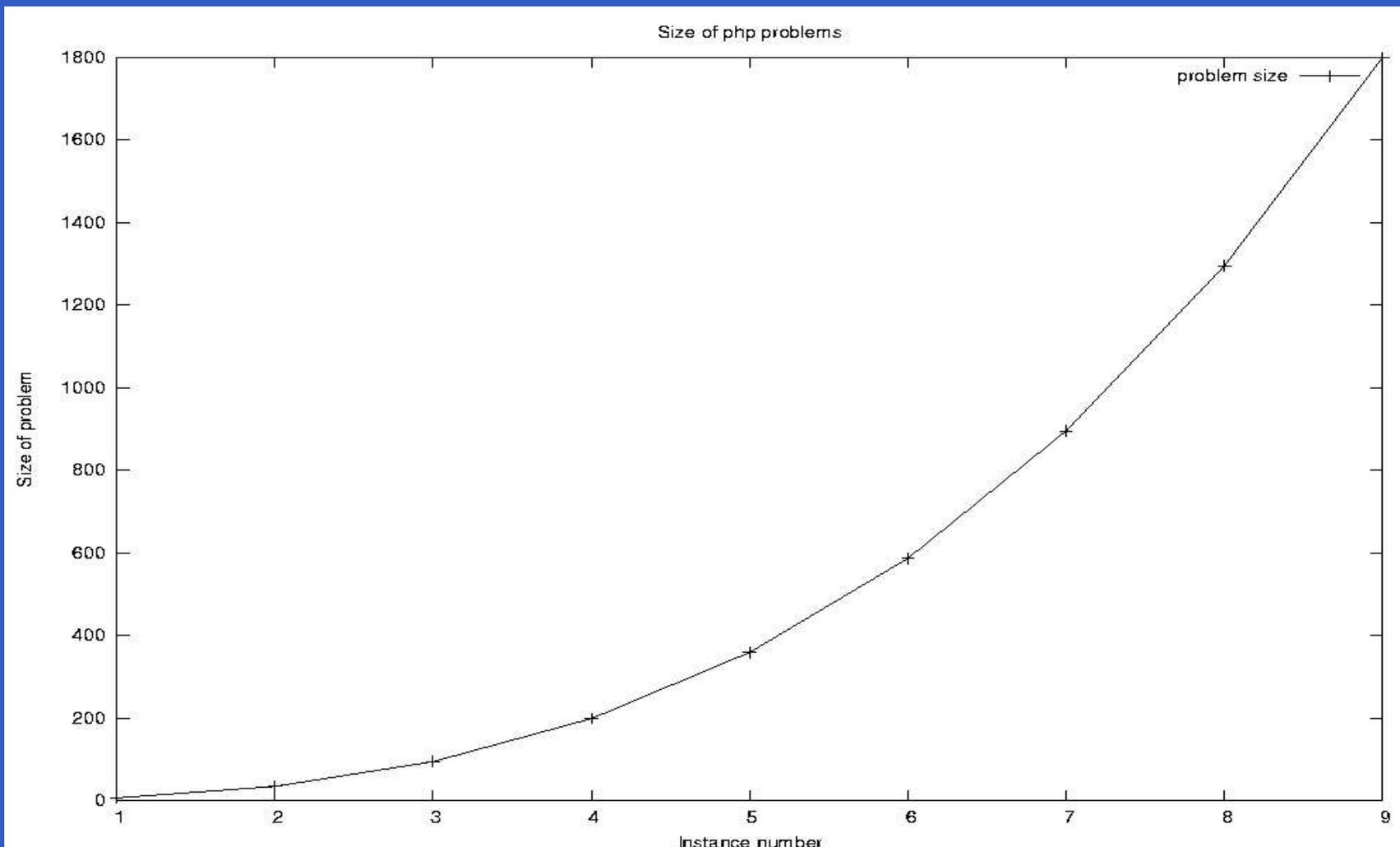
- Árvore de prova (vai aumentando mas partes podem ser descartadas)
- Fórmulas e fórmulas assinaladas como árvores: facilitar verificação e descoberta de subfórmulas
- Fábricas de fórmulas e fórmulas assinaladas: evitar desperdício de memória
- Pilha de ramos abertos

Famílias de problemas difíceis

Algumas famílias de problemas utilizadas para testar a complexidade computacional de provadores:

- A família PHP (Princípio do Escaninho) é uma família de tautologias muito difíceis de provar em alguns sistemas de prova. Existe uma prova polinomial em cálculo de seqüentes com corte.
- fórmulas de Statman, Γ , H, U, Tseitin

Família PHP



PHP₄ - Tamanho do problema

Tamanho do problema: 199

- Arquivo: 598 B
- Variáveis proposicionais: 20
- Fórmulas compostas: 99
- Conectivos: 3
- Complexidade da fábrica de fórmulas: 3759
- Complexidade da fábrica de fórmulas assinaladas =
Tamanho do problema: 199

PHP₄ - Tamanho da prova

Tamanho da prova: 4959

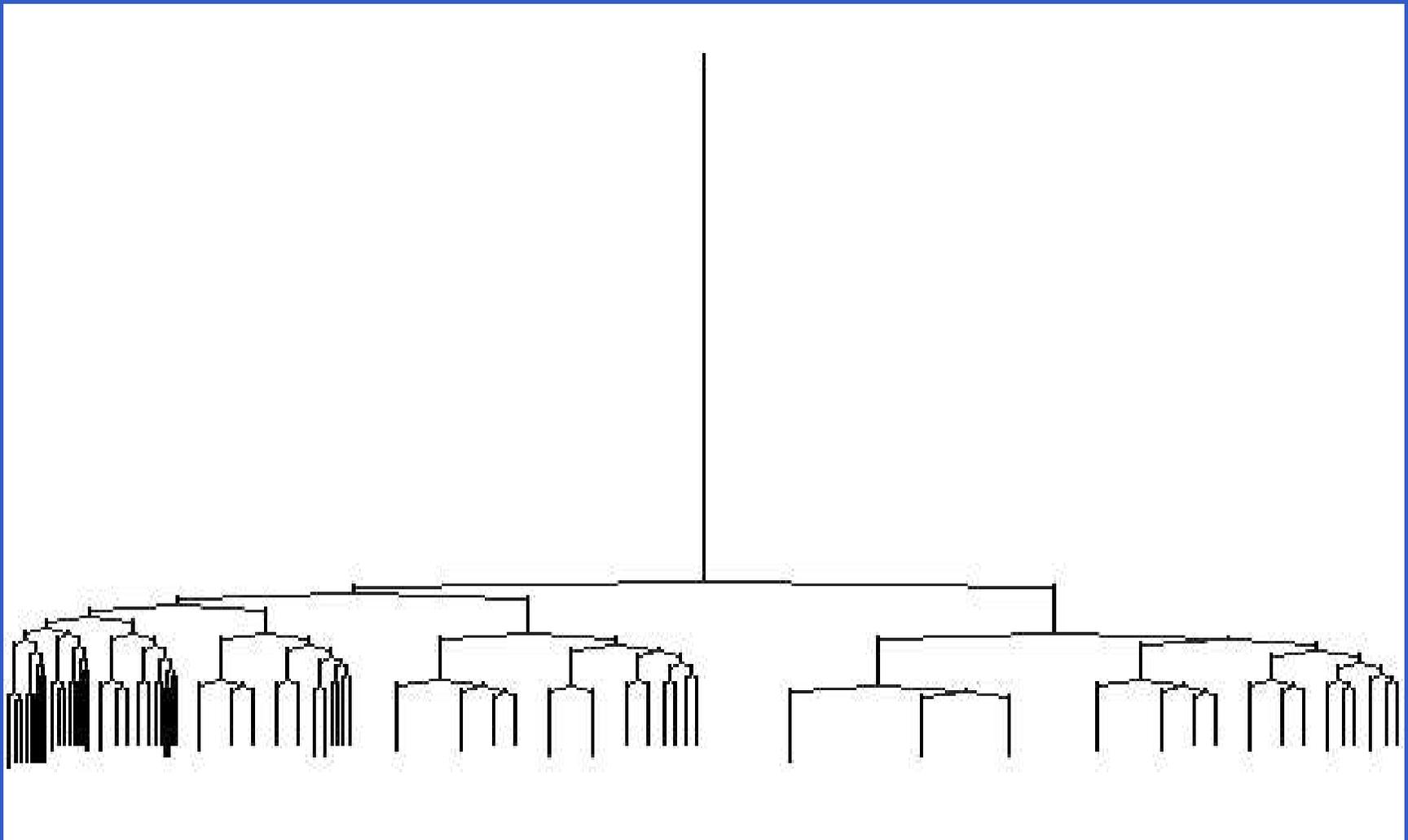
- Arquivo: 33,8 KB (básica) ou 338,2KB (completa)
- Variáveis proposicionais: 20
- Número de ocorrências de fórmulas assinaladas: 1127
- Altura da árvore: 10
- Fórmulas compostas: 99
- Conectivos: 3

PHP₄ - Tamanho da prova

Tamanho da prova: 4959

- Complexidade da fábrica de fórmulas: 3767
- Complexidade da fábrica de fórmulas assinaladas: 3786
- Regras utilizadas: 1679
 - Regras simples: 115
 - Regras simplificadoras: 1493
 - Regra PB: 71
- Ramos: 143

Árvore de prova de PHP₄



Resultados obtidos

MSTP

family	instance	time	signed formulas	height	size
H	6	10.693	605	5	33313
Γ	7	0.143	53	0	440
Γ	100	3.788	805	0	2205
Statman	6	0.302	33	0	440
Statman	21	3.013	258	0	13425
PHP	4	3.089	1127	10	4959
PHP (in clausal form)	4	6.497	2101	10	10860

Resultados obtidos

WDTP

family	instance	time	signed formulas	height
H	6	101.803705	1953	132
Γ	7	4.319546	2917	12
Statman	6	9.807499	5391	16
PHP	4	4.247875	483	26
PHP	5	229.452014	5409	53
PHP (in clausal form)	4	17.826073	2147	33

Conclusão

- Provas menores foram obtidas
- Tempo menor mesmo com linguagem mais lenta
- Melhor modularização das estratégias

Trabalhos futuros

- Implementação de novas estratégias
- Utilização de outras famílias de problemas e da biblioteca SATLIB
- Implementação de regras e estratégias para lógicas de inconsistência formal

Dúvidas, perguntas, sugestões, ...

- Dúvidas?
- Perguntas?
- Sugestões?
- Correções?

Um Proveedor de Teoremas Multi-Estratégia

Adolfo Gustavo Serra Seca Neto e Marcelo Finger

Instituto de Matemática e Estatística - USP

Artigos, relatórios, executáveis:

<http://www.ime.usp.br/~adolfo>

Contato:

adolfo@ime.usp.br

Grato pela atenção!