Um Provador 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

Provadores 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)

Provadores Automáticos de Teoremas

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

Entrada para PAT's

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

Exemplo de entrada: PHP_2

- Fórmula:
  $\mathtt{F}\,(((p_{0,0} \vee p_{0,1}) \wedge (p_{1,0} \vee p_{1,1}) \wedge (p_{2,0} \vee p_{2,1})) \rightarrow ((p_{0,0} \wedge p_{1,0}) \vee (p_{0,0} \wedge p_{2,0}) \vee (p_{1,0} \wedge p_{2,0}) \vee (p_{0,1} \wedge p_{1,1}) \vee (p_{0,1} \wedge p_{2,1}) \vee (p_{1,1} \wedge p_{2,1})))$
- Variáveis proposicionais: 6
- Fórmulas: 18
- Fórmulas assinaladas: 1
- Complexidade: 35

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

$\mathtt{F}\,(A \to (B \to (C \to A)))$

$\mathtt{T}\,A$

$\mathtt{F}\,B \to (C \to A)$

$\mathtt{T}\,B$

$\mathtt{F}\,C \to A$

$\mathtt{T}\,C$

$\mathtt{F}\,A$

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)

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 efi ciente que os TA's e permite uma diversidade maior de estratégias

Sistema KE - regras

$\dfrac{T\ A \vee B \quad F\ A}{T\ B}$	$\dfrac{F\ A \wedge B \quad T\ A}{F\ B}$	$\dfrac{T\ A \rightarrow B \quad T\ A}{T\ B}$	$\dfrac{T\ \neg A}{F\ A}$
$\dfrac{T\ A \vee B \quad F\ B}{T\ A}$	$\dfrac{T\ A \wedge B}{T\ A \quad T\ B}$	$\dfrac{T\ A \rightarrow B \quad F\ B}{F\ A}$	$\dfrac{F\ \neg A}{T\ A}$
$\dfrac{F\ A \vee B}{F\ A \quad F\ B}$	$\dfrac{F\ A \wedge B}{T\ B \quad F\ A}$	$\dfrac{F\ A \rightarrow B}{T\ A \quad F\ B}$	$\dfrac{}{T\ A \mid F\ A}$

Prova mínima em KE

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

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

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

$\mathtt{T}\,A$

$\mathtt{F}\,C1 \rightarrow C2 \rightarrow C3 \rightarrow A$

$\mathtt{T}\,C1$

$\mathtt{F}\,C2 \rightarrow C3 \rightarrow A$

$\vdots$

Prova mínima em KE

$\vdots$

$\mathtt{T}\, C2$

$\mathtt{F}\, C3 \rightarrow A$

$\mathtt{T}\, C3$

$\mathtt{F}\, A$

$\mathtt{x}$

Prova com passos desnecessários

$\mathtt{F}\,(A \to B1 \to B2 \to B3) | (A \to C1 \to C2 \to C3 \to A)$

$\mathtt{F}\,(A \to B1 \to B2 \to B3)$

$\mathtt{F}\,(A \to C1 \to C2 \to C3 \to A)$

$\mathtt{T}\,A$

$\mathtt{F}\,C1 \to C2 \to C3 \to A$

$\mathtt{F}\,B1 \to B2 \to B3(*)$

$\mathtt{T}\,C1$

$\mathtt{F}\,C2 \to C3 \to A$

$\vdots$

Prova com passos desnecessários

$\vdots$

T $B1(*)$

F $B2 \rightarrow B3(*)$

T $C2$

F $C3 \rightarrow A$

T $B2(*)$

F $B3(*)$

T $C3$

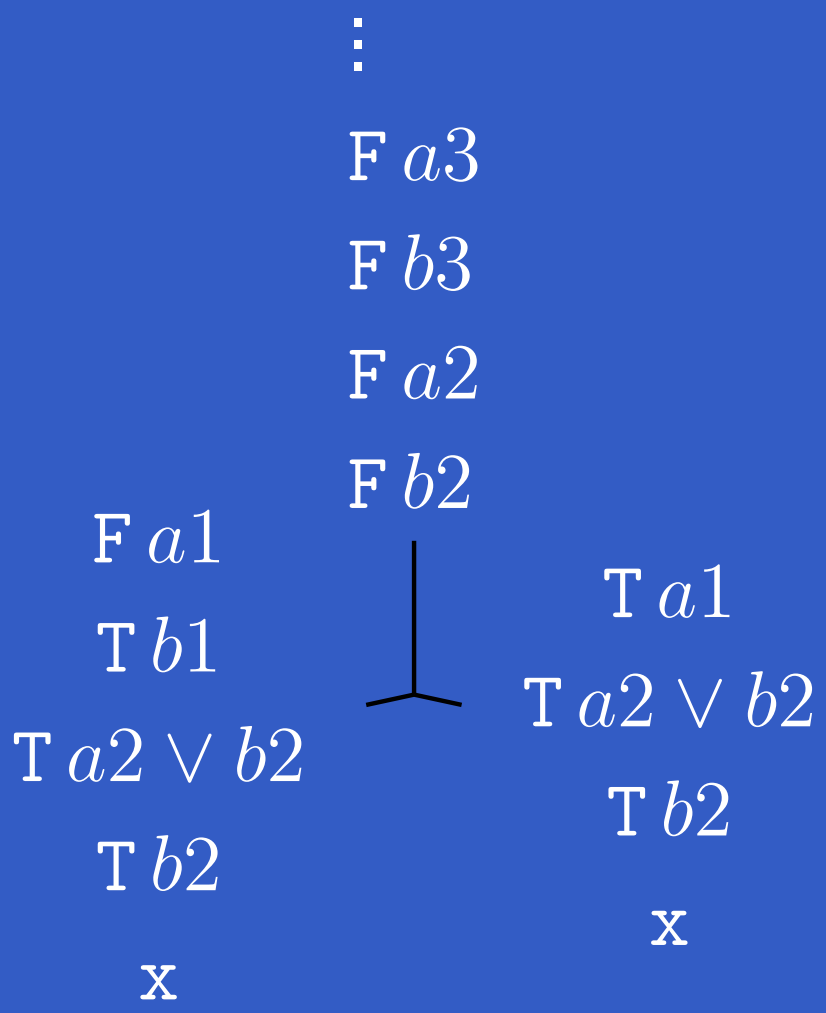
F A

x

Exemplo de uso da regra PB

$$\mathtt{T}\,(a1 \vee b1)$$
$$\mathtt{T}\,(a1 \rightarrow (a2 \vee b2))$$
$$\mathtt{T}\,(b1 \rightarrow (a2 \vee b2))$$
$$\mathtt{T}\,(a2 \rightarrow (a3 \vee b3))$$
$$\mathtt{T}\,(b2 \rightarrow (a3 \vee b3))$$
$$\mathtt{F}\,(a3 \vee b3)$$
$$\vdots$$

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

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

$$\frac{X\,\Phi(\top \vee A)}{X\,\Phi(\top)}\ (X \vee \top) \qquad \frac{X\,\Phi(\bot \vee A)}{X\,\Phi(A)}\ (X \vee \bot)$$

Provador 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: provador que escolhe estratégia de acordo com características do problema

O provador multi-estratégia (MSTP)

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

O provador multi-estratégia (MSTP)

- O provador recebe um problema e retorna uma prova.
- Uma prova contém uma árvore de prova (fechada ou não).
- Um provador utiliza um método e uma estratégia.
- O método fornece o conjunto de regras.
- A estratégia executa o algoritmo que aplica as regras.

O provador multi-estratégia (MSTP)

Estrutura básica:

- Lógica: formulasNew e signedFormulasNew (pcl.jar)
- Regras: rules (rules.jar)
- Provador: Prover, Method, Strategy, Proof, ProofTree, ClassicalProofTree
- Fachada: TableauFacade
- profiler aspects (medir tempo, salvar provas e etc.)

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. Verficar 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 prem. simplificadoras

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

Regras de duas prem. simplificadoras

- Para cada fórmula assinalada no ramo, verifca 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 tablô 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

Segunda estratégia implementada

- Motivação: estratégia anterior não prova PHP_5 por falta de memória.
- Árvore de prova contém fórmulas que não serão mais necessárias apenas para a geração do arquivo contendo a prova.
- Árvore de prova contém origem de cada fórmulas (regra e premissas)
- Referências a subfórmulas de cada fórmula em cada ramo são salvas em FormulaReferenceClasicalProofTree.

Segunda estratégia implementada

Melhorias:

- Descartar ramos fechados (tira da memória)
- Não salvar a origem das fórmulas (regra e premissas)
- Pesquisar as referências a uma fórmula em vez de guardá-las na memória

Segunda estratégia implementada

- OptimizedClassicalProofTree
- NullClosedProofTree
- CloseableProofTree
- ReferenceFinder
- MemorySaverStrategy
- alterações em ProofTree, ClassicalProofTree, Prover, Strategy, . . .

Segunda estratégia implementada

Resultados:

- Conseguiu provar PHP_5 e PHP_6
- Arquivo-prova bem simplificado (não evita repetições, não mantém ordem natural)

Limitações adicionais (além da ausência da origem):

- Arquivo-prova bem simplificado (não evita repetições, não mantém ordem natural)

Multiplas estratégias

- implementar uma estratégia significa implementar algumas classes e aspectos
- possivelmente modificar classes e aspectos já existentes
- uma vez implementada a nova estratégia, pode-se compará-la com as outras na mesma implementação

Uso de padrões de projeto

- Singleton: NullClosedProofTree
- Composite
- Flyweight (e Factory)
- Strategy
- Facade

Uso de aspectos

- Para fazer profiling e depuração. Ex.: medições (dados da execução, regras utilizadas, salvar provas) e nomes dos ramos.
- Para separar requisitos de uma mesma classe. Ex.: tamanho da árvore de prova.
- Para implementar requisitos transversais. Ex.: complexidade de fórmulas (assinaladas) e suas fábricas.

Uso de aspectos

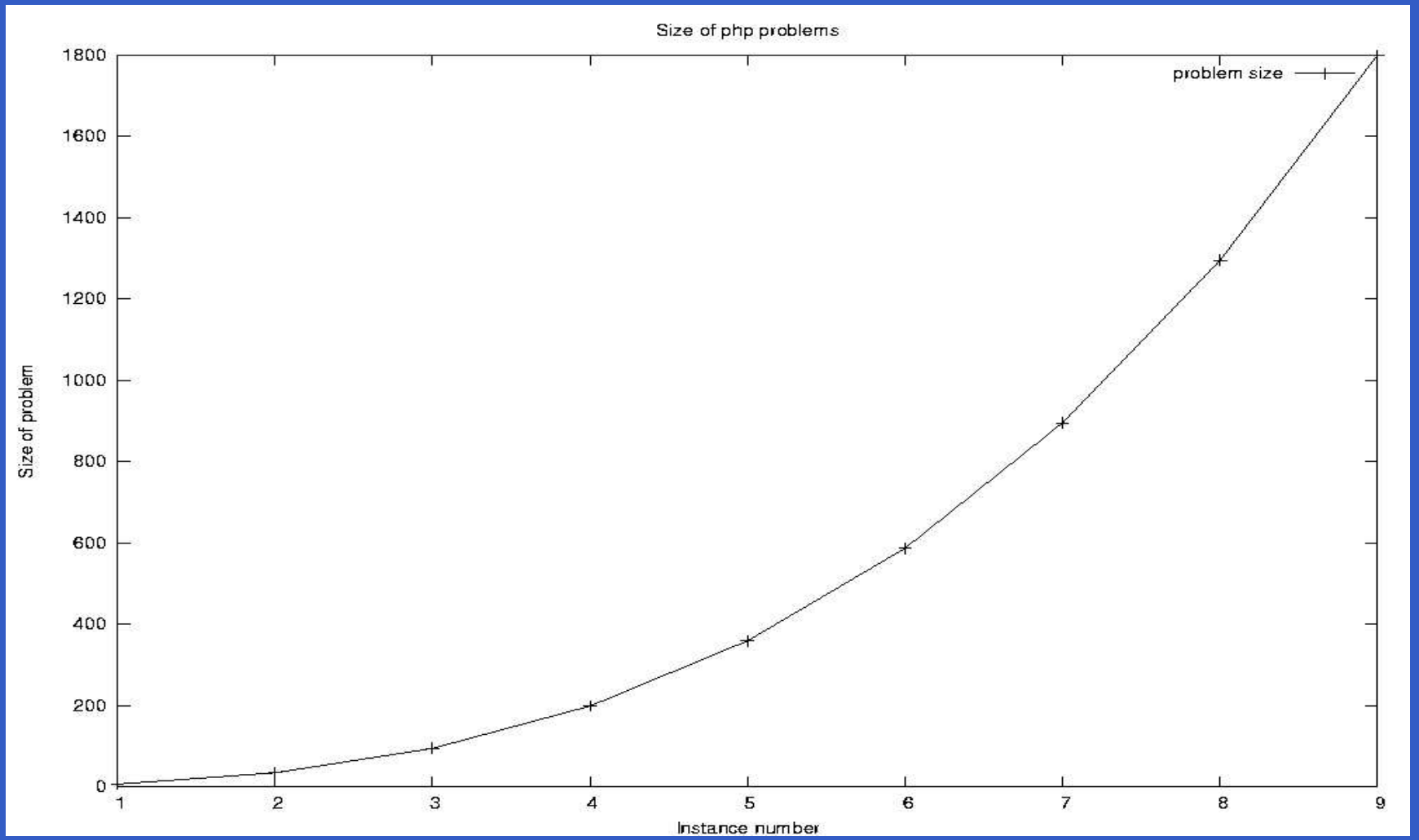
- Para testar novas estratégias rapidamente
- Para modificar várias estratégias de uma vez
- Para construir estratégias em tempo de execução a partir de características de estratégias definidas em aspectos: necessita de suporte a entrelaçamento dinâmico (ou de uma simulação deste através de padrões de codificação)

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

Famíla PHP


Size of php problems
problem size
Size of problem
Instance number
0
200
400
600
800
1000
1200
1400
1600
1800
1
2
3
4
5
6
7
8
9

PHP_4 - 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

PHP_4 - Tamanho da prova

Tamanho da prova: 4959

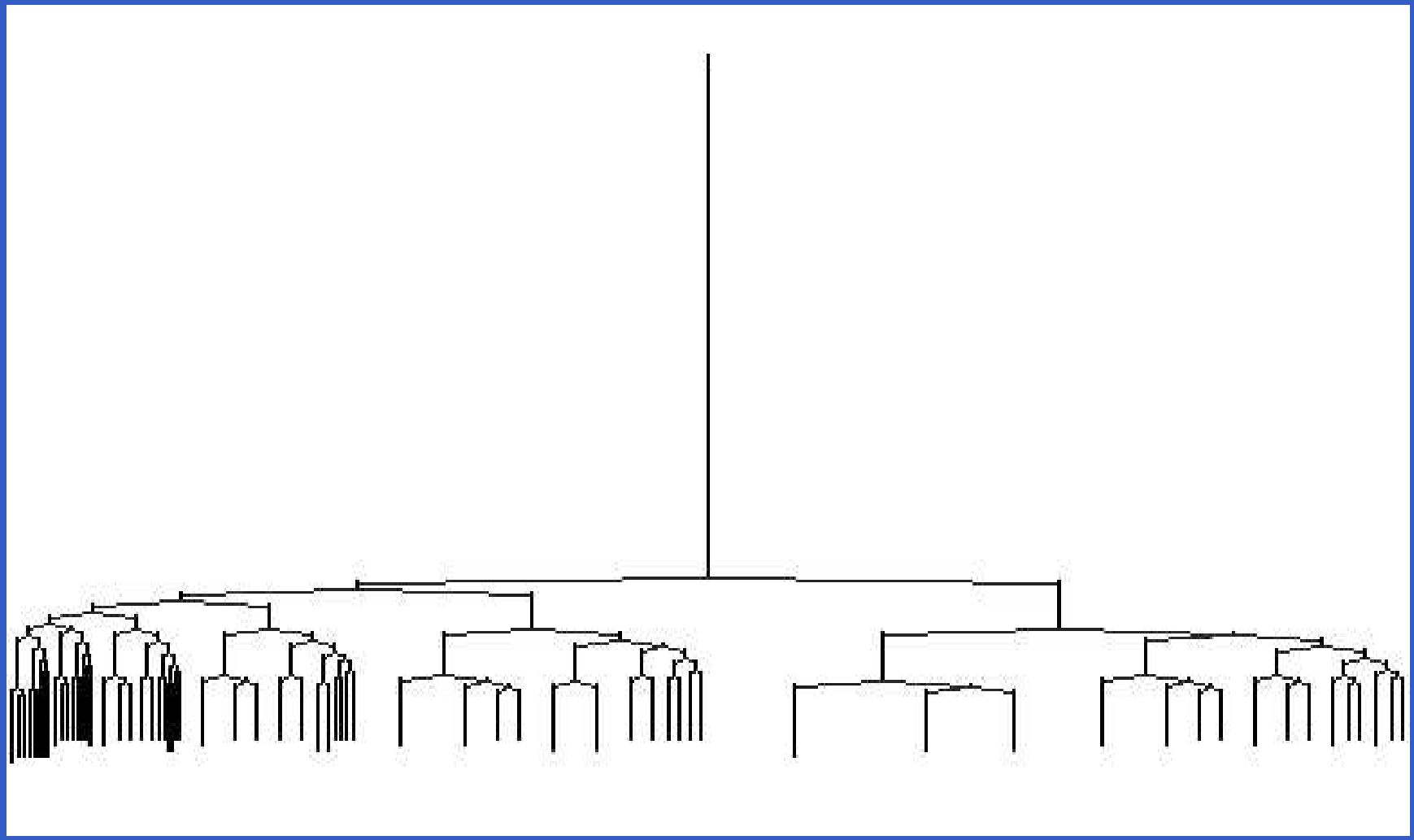
- Arquivo: 33,8 KB (básica) ou 338,2KB (completa)
- Variáveis proposicionais: 20
- Número de ocorrências de fórmulas assinaladas: 1127
- Altura da árvore: 10
- Fórmulas compostas: 99
- Conectivos: 3

PHP_4 - 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 PHP_4



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

Resultados obtidos

MSTP c/SS

family	instance	time	signed formulas	height	size
H	6	1.222	605	5	33313
Γ	7	0.148	53	0	440
Γ	100	2.831	805	0	2205
Statman	6	0.119	33	0	440
Statman	21	0.641	258	0	13425
PHP	4	1.078	1127	10	4959
PHP (in clausal form)	4	2.514	2101	10	10860

Resultados obtidos

MSTP c/MSS

family	instance	time	signed formulas	height	size
H	6	0.979	605	5	33313
Γ	7	0.308	53	0	440
Γ	100	22.299	805	0	2205
Statman	6	0.301	33	0	440
Statman	21	0.583	258	0	13425
PHP	4	0.911	1127	10	4959
PHP	5	21.667	?	?	?
PHP	6	638.985	?	?	?
PHP (in clausal form)	4	5.551	2101	10	10860
PHP (in clausal form)	5	95.899	?	?	?

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
- Modularização de estratégias e características de estratégias
- Escolha em tempo de execução das carcaterísticas das estratégias

Dúvidas, perguntas, sugestões, . . .

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

Um Provador de Teoremas Multi-Estratégia

Adolfo Gustavo Serra Seca Neto e Marcelo Finger

Instituto de Matemática e Estatística - USP

FIM

Artigos, relatórios, executáveis:

http://www.ime.usp.br/∼adolfo

Contato:

adolfo@ime.usp.br

Grato pela atenção!