Fast Equational Reasoning

with WALDMEISTER

Thomas Hillenbrand

Max-Planck-Institut für Informatik
Saarbrücken
Aim of this Talk

- RTA organizers:
  “... would be nice to show how a combination of the theory of rewriting, implementation techniques, heuristics, ideas ... whatever else ... lead to a design of the fastest equational reasoner in the world”

- Some evidence of “fastest” from performance in the CADE ATP System Competitions. A.D. 2007 (100 problems attempted):

<table>
<thead>
<tr>
<th></th>
<th>WM</th>
<th>VAMPIRE</th>
<th>E</th>
<th>OTTER</th>
<th>METIS</th>
<th>EQUINOX</th>
<th>GEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>91</td>
<td>63</td>
<td>59</td>
<td>27</td>
<td>15</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>av. time</td>
<td>18.2</td>
<td>42.3</td>
<td>16.7</td>
<td>21.6</td>
<td>38.3</td>
<td>13.4</td>
<td>255.8</td>
</tr>
</tbody>
</table>

- What are the underlying concepts?
Outline

- Foundations
- Prover engineering
- Controlling redundancy
- Applications
I Foundations
**Equational Logic**

- **Example:** group axiomatization

  \[
  E : \ (x + y) + z = x + (y + z) \quad x + 0 = x \quad x + (-x) = 0
  \]

  *Word problem:* Does \( E \models x = - - x \) hold?
  (Birkhoff 1935): replace *equals by equals*

- **Confluent** and *terminating* theory presentation:
  Apply equations *non-deterministically* and in *one direction* only
  Word problem *decidable* by computation of *normal forms*

- If terminating: confluence = *local* confluence (Newman 1942),
  effective test via *Critical Pair Lemma* (Knuth, Bendix 1970):
  Check if critical pairs rewrite into tautologies
Completion

- In the **negative** case:
  - **enrich** presentation with rewritten critical pairs
  - perform mutual **simplification**
  - **iterate** the procedure!

  \{ essence of Knuth-Bendix completion \}

- **Fails** if non-orientable equations encountered
  
  **Ordered completion** takes orientable instances into account, produces **ground confluent** system in the limit (Lankford 1975)

- Limit normal form reached in **finite** approximation already
  
  Semi-decision procedure for word problem with **drastically reduced** search space (Hsiang, Rusinowitch 1987)
Ordered Completion

- **Proof-theoretic** framework (Bachmair, Dershowitz, Hsiang 1986): Completion as *transformation of proofs*, contained in well-founded *proof ordering* where *rewrite proofs* are minimal. Proof steps weighted according to

  \[ s \xleftarrow{u\Rightarrow m} t \xrightarrow{} (\{s\}, u, m, t) \text{ if } s \succ t \]

- Deduction of new facts must ensure *fairness*: eventually smaller proof for every persistent *ground peak* \( s \leftarrow t \rightarrow u \) in \( \Sigma^e \). Equation *redundant* if every ground instance has smaller proof.

- **WALDMEISTER** as an implementation of ordered completion: performs *fully automated* proof search, returns *proof log* in case of success . . .
new rule: 1  + (x1, 0)  ->  x1
new rule: 2  + (x1, -(x1))  ->  0
new rule: 3  + (+ (x1, x2), x3)  ->  + (x1, +(x2, x3))
new rule: 4  + (x1, +(0, x2))  ->  +(x1, x2)
new rule: 5  + (x1, -(0))  ->  x1
new rule: 6  + (x1, +(0, -(x1), x2))  ->  +(0, x2)
new rule: 7  + (0, -(x1))  ->  x1
new rule: 8  + (0, -(-(x1)))  ->  +(x1, x2)
remove rule: 7
new rule: 9  + (0, x1)  ->  x1
remove rule: 4
simplify rhs of rule: 6
new rule: 10  -(0)  ->  0
remove rule: 5
new rule: 11  -(-(x1))  ->  x1
remove rule: 8
joined goal: 1  c  ?=  -(-(c))  to  c

Proved Goals:
No. 1: c  ?=  -(-(c))  joined, current: c = c
1 goal was specified, which was proved.
Waldmeister states: Goal proved.
Consider the following set of axioms:

Axiom 1: \( x + 0 = x \)
Axiom 2: \( x + (-x) = 0 \)
Axiom 3: \( (x + y) + z = x + (y + z) \)

This theorem holds true:

Theorem 1: \( x = - - x \)

Proof:

Lemma 1: \( 0 + (- - x) = x \)

\[
0 + (- - x) = \quad \text{by Axiom 2 RL}
\]
\[
(x + (-x)) + (- - x) = \quad \text{by Axiom 3 LR}
\]
\[
x + ((-x) + (- - x)) = \quad \text{by Axiom 2 LR}
\]
\[
x + 0 = \quad \text{by Axiom 1 LR}
\]
\[
x = \quad \text{by Axiom 1 LR}
\]

Lemma 2: \( x + (- - y) = x + y \)

\[
x + (- - y) = \quad \text{by Axiom 1 RL}
\]
\[
(x + 0) + (- - y) = \quad \text{by Axiom 3 LR}
\]
\[
x + (0 + (- - y)) = \quad \text{by Lemma 1 LR}
\]
\[
x + y = \quad \text{by Lemma 1 LR}
\]

Lemma 3: \( 0 + x = x \)

\[
0 + x = \quad \text{by Lemma 2 RL}
\]
\[
0 + (- - x) = \quad \text{by Lemma 1 LR}
\]
\[
x = \quad \text{by Lemma 3 LR}
\]

Theorem 1: \( x = - - x \)

\[
x = \quad \text{by Lemma 3 RL}
\]
\[
0 + x = \quad \text{by Lemma 2 RL}
\]
\[
0 + (- - x) = \quad \text{by Lemma 3 LR}
\]
\[
x = \quad \text{by Lemma 3 LR}
\]
Calculus and Proof Procedure

- Ordered / unfailing completion: given as set of *calculus rules*
  
  \[ l = r \quad s[l'] = t \]
  
  critical pairing

  contracting: rewrite-based simplification rules

- Additional *control constraint*: fairness
  
  *Parameter*: reduction ordering

- How to turn this into a *deterministic algorithm*?
  
  Common solutions:
  - given-pair algorithm (Wos, Carson, Robinson 1964)
  - Huet’s algorithm (Huet 1981)
  - *given-clause* algorithm (Overbeek 1971)
Given-clause Algorithm

- Approach: incrementally precompute all expansion steps assess candidate equations heuristically by weighting function $\varphi$

- Active facts $A$ for rewriting and superposition
  Passive facts $P$: critical pairs descending from $A$

\[ s=t: \varphi(s=t) \text{ min.} \]

$A$ $\rightarrow\leftarrow$ $P$

$\text{CP}^>(s=t, A)$
FUNCTION \textsc{waldmeister}(\Sigma, \mathcal{E}, \mathcal{C}, >, \varphi) : BOOL

1: \quad (\mathcal{A}, \mathcal{P}) := (\emptyset, \mathcal{E})

2: \quad \textbf{WHILE} \neg\text{trivial}(\mathcal{C}) \land \mathcal{P} \neq \emptyset \ \textbf{DO}

3: \quad e := \min_{\Phi}(\mathcal{P}); \quad \mathcal{P} := \mathcal{P} \setminus \{e\}

4: \quad e := \text{Normalize}_{\mathcal{A}}(e)

5: \quad \textbf{IF} \neg\text{redundant}(e) \ \textbf{THEN}

6: \quad (\mathcal{A}, \mathcal{P}_1) := \text{Interred}^>(\mathcal{A}, e)

7: \quad \mathcal{A} := \mathcal{A} \cup \{\text{Orient}^>(e)\}

8: \quad \mathcal{P}_2 := \text{CP}^>(e, \mathcal{A})

9: \quad \mathcal{P} := \text{Update}(\mathcal{P} \cup \mathcal{P}_1 \cup \mathcal{P}_2)

10: \quad \mathcal{C} := \text{Normalize}_{\mathcal{A}}(\mathcal{C})

11: \quad \textbf{END}

12: \quad \textbf{END}

13: \quad \textbf{RETURN} \ \text{trivial}(\mathcal{C})
**Proof Procedure**

**FUNCTION** \( \text{WALDMEISTER}(\Sigma, E, C, >, \varphi) : BOOL \)

1: \((A, P) := (\emptyset, E)\)
2: \textbf{WHILE} \(\neg\text{trivial}(C) \land P \neq \emptyset\) \textbf{DO}
3: \(e := \min_\varphi(P); P := P \setminus \{e\}\)
4: \(e := \text{Normalize}_A(e)\)
5: \textbf{IF} \(\neg\text{redundant}(e)\) \textbf{THEN}
6: \((A, P_1) := \text{Interred}^>(A, e)\)
7: \(A := A \cup \{\text{Orient}^>(e)\}\)
8: \(P_2 := \text{CP}^>(e, A)\)
9: \(P := \text{Normalize}_A(P \cup P_1 \cup P_2)\) \hspace{1cm} \textbf{Otter loop – eager}
10: \(C := \text{Normalize}_A(C)\)
11: \textbf{END}
12: \textbf{END}
13: \textbf{RETURN} \text{trivial}(C)
FUNCTION WALDMEISTER(\(\Sigma, E, C, >, \varphi\)) : BOOL

1: \((A, P) := (\emptyset, E)\)
2: WHILE \(\neg\text{trivial}(C) \land P \neq \emptyset\) DO
3: \(e := \min_\varphi(P); \ P := P \setminus \{e\}\)
4: \(e := \text{Normalize}_A(e)\)
5: IF \(\neg\text{redundant}(e)\) THEN
6: \((A, P_1) := \text{Interred}^>(A, e)\)
7: \(A := A \cup \{\text{Orient}^>(e)\}\)
8: \(P_2 := \text{CP}^>(e, A)\)
9: \(P := P \cup \text{Normalize}_A(P_1 \cup P_2)\)  \hspace{1cm} \text{DISCOUNT loop – lazy}
10: \(C := \text{Normalize}_A(C)\)
11: END
12: END
13: RETURN trivial(C)
II Prover Engineering
Introduction

- For actual *realization* of proof procedure:
  Design / adapt appropriate *algorithms* and *data structures*!
  Functionality, time efficiency, space efficiency

- *Time-space* tradeoffs frequent in CS
  Additionally: take modern *memory hierarchies* into account!
  Can *quickly* access only a *small* part of memory

- *Entities* to represent: active facts, passive facts, conjecture

- *Control parameters* of proof procedure:
  reduction ordering and weighting function
  Pragmatic approach of *automating control*
Representing the Active Facts

- Essentially: incrementally constructed *data base* of term( pair)s
  Inferencing, simplifying = *complex retrieval* from data base

- *Retrieval conditions*: more general / unifiable / less general terms
  Major part of system’s work: *normalizing* new critical pairs, requires retrieval of generalizations

- Inference rate *soon sharply decreases* if retrieval handled 1:1
  “Performance degradation” (Wos 1992)

- Remedy: retrieval in *set-based* fashion
  Process at a time one query against a *compiled* data base!
  “*Term indexing*”, indispensable in today’s ATP systems
Discrimination Trees (1)

- Term as *string* of its symbols, indexed in *trie* data structure
  Sharing of *common prefixes* (Christian 1989)

- Example: Index for term set
  
  \[
  f(x_1, x_1) \\
  f(x_1, b) \\
  f(a, g(x_1)) \\
  f(g(x_1), g(x_2)) \\
  f(g(b), a)
  \]

- Retrieval typically via *backtracking* due to *non-determinism* in descent
Discrimination Trees (2)

- Optimization: *collapse* subtrees with only one leaf node
  May cut away *more than half* of the nodes
  Data structure *more compact*, retrieval *faster*

- Query terms traversed "*from left to right*"
  Hard-wired into term representation: ...
Discrimination Trees (2)

- Optimization: *collapse* subtrees with only one leaf node
  May cut away *more than half* of the nodes
  Data structure *more compact*, retrieval *faster*

- Query terms traversed "*from left to right*"
  Hard-wired into term representation:

  *Flatterms* (Christian 1989) instead of *tree-like*
Which Indexing Technique is Optimal?

- **Complexity analysis** of indexing techniques **difficult** (Graf 1996)

- **COMPIT** initiative (Nieuwenhuis, H., Riazanov, Voronkov 2001): Compare *implementations* of different techniques on *benchmarks* corresponding to real runs of real provers

- Speed in 2000: code trees : descr. trees : context trees
  
  1.91 : 1.37 : 1.00

- Participants have *improved* their implementations since DTs: nearly twice as fast just by more compact node format

- Careful coding counts!
Representing the Passive Facts

- $\mathcal{P}$ ordered under $\varphi$: functionality of priority queue

- Typically $|\mathcal{P}|$ exceeding $|\mathcal{A}|$ by three orders of magnitude. Space can become a problem! Standard solution: discard heavy equations – completeness lost

- **DISCOUNT loop**: no rewriting on passive facts! Successively more compact representations:
  
  - flattened terms: $f - x_1 - f - a - x_2 = f - x_1 - x_2$
  
  - string terms: $f \ x_1 \ f \ a \ x_2 \ f \ x_1 \ x_2$
  
  - implicit: $<s[l']_p = t, l = r>$
Space Behaviour over Time

![Graph showing space requirements over time for activated facts. The graph plots the number of activated facts against space requirements, with lines representing different ROB001-1 categories: flatterms, stringterms, overlap, and without P.](image)
Towards the WALDMEISTER Loop

- **Group together** elements generated during *same* loop iteration: themselves *ordered* by \( \varphi \), occasional removal of *lightest* element

- If *re-generation* + *re-normalization* available and weights unique: only need to store the *next minimal weight* retrievable from group! *Priority queue* on top of these entries as before

- Crucial issue in *reproduction*: need *same weights*, hence *same normal forms*
  Nice: *whole history* of \( A \) fits into *one DT* with *age constraints*
  Prerequisite for practicality: *cache* for lightweight entries

- All in all: space for \( \mathcal{P} \) *linear* in \( |A| \). *Laziness works!*
  Besides: *proof objects* for free, *parallelization* possible
Space Behaviour over Time (revisited)

![Graph showing space requirements over the number of activated facts.](image)

- **ROB001-1**
  - `flatters`: Red line
  - `stringterms`: Blue line
  - `overlap`: Green line
  - `without P`: Magenta line
  - `NEW`: Cyan line

**Number of activated facts**
- 0
- 1000
- 2000
- 3000
- 4000
- 5000

**Space requirements**
- 0 MB
- 250 MB
- 500 MB
- 750 MB
- 1 GB

Th. Hillenbrand
Representing the Conjecture

- Instead of *termpair*, consider *sets of rewrite successors* in order to join left- and right-hand side earlier
- *Example*: GRP141-1 when 0 rewrite rules derived

\[ \mathcal{U} \quad \mathcal{U} \]

\[ \mathcal{V} \quad \mathcal{V} \]
Representing the Conjecture

- Instead of *termpair*, consider *sets of rewrite successors* in order to join left- and right-hand side earlier
- **Example:** GRP141-1 when 2 rewrite rules derived
Representing the Conjecture

- Instead of *termpair*, consider *sets of rewrite successors* in order to join left- and right-hand side earlier
- *Example:* GRP141-1 when 13 rewrite rules derived
Representing the Conjecture

- Instead of *termpair*, consider *sets of rewrite successors* in order to join left- and right-hand side earlier

- Example: GRP141-1 when 19 rewrite rules derived
Representing the Conjecture

- Instead of *term pair*, consider *sets of rewrite successors* in order to join left- and right-hand side earlier

- *Example*: GRP141-1 when 30 rewrite rules derived
Benefit Derived from Successor Sets

- Proofs are found
  - in many cases with *less steps* of saturating the axiomatization
  - at least with *no more* steps

- Some proofs *only* found with enlarging

- Focus of completion-based proving *slightly shifts*
  from axioms to conjecture

- Extension: consider (some) rewrite *predecessors* as well
  Danger of combinatorical explosion – strict limit needed
Automating Control: Weighting Function

- Comparison of *weighting functions* \( \varphi \) in various domains

<table>
<thead>
<tr>
<th>t/s [SPARC]</th>
<th>addweight</th>
<th>gtweight</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOO003-2</td>
<td>&gt;300</td>
<td>0.1</td>
</tr>
<tr>
<td>BOO007-2</td>
<td>&gt;300</td>
<td>81.8</td>
</tr>
<tr>
<td>BOO008-4</td>
<td>61.1</td>
<td>7.0</td>
</tr>
<tr>
<td>LCL153-1</td>
<td>2.1</td>
<td>&gt;300</td>
</tr>
<tr>
<td>LCL154-1</td>
<td>2.0</td>
<td>&gt;300</td>
</tr>
<tr>
<td>LCL155-1</td>
<td>1.2</td>
<td>&gt;300</td>
</tr>
<tr>
<td>( \Sigma ) Boolean</td>
<td>22 / 29</td>
<td>29 / 29</td>
</tr>
<tr>
<td></td>
<td>25.4</td>
<td>4.5</td>
</tr>
<tr>
<td>( \Sigma ) Wajsberg</td>
<td>21 / 25</td>
<td>17 / 25</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.9</td>
</tr>
</tbody>
</table>

- Must employ *different weighting functions* on different structures!
Automating Control: Reduction Ordering

- **Lexicographic path** ordering: lifts operator precedence to terms
- **Knuth-Bendix** ordering: orders terms according to their length

<table>
<thead>
<tr>
<th>t/s [SPARC]</th>
<th>LPO</th>
<th>KBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL063-4</td>
<td>223.0</td>
<td>0.0</td>
</tr>
<tr>
<td>COL063-6</td>
<td>&gt;300</td>
<td>0.0</td>
</tr>
<tr>
<td>COL064-6</td>
<td>&gt;300</td>
<td>0.0</td>
</tr>
<tr>
<td>(\sum) BT fragment</td>
<td>21 / 27</td>
<td>25 / 27</td>
</tr>
<tr>
<td></td>
<td>16.6</td>
<td>0.5</td>
</tr>
<tr>
<td>(\sum) non-associative rings</td>
<td>21 / 38</td>
<td>11 / 38</td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td>1.4</td>
</tr>
<tr>
<td>(A \succ C \succ * \succ - \succ + \succ 0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\sum) lattice-ordered groups</td>
<td>98 / 102</td>
<td>90 / 102</td>
</tr>
<tr>
<td></td>
<td>12.7</td>
<td>23.8</td>
</tr>
</tbody>
</table>

- Must employ **different orderings** on different structures!
Control Component (1)

- **Recognize** known axiomatizations within input specification $\mathcal{E}$
- **Stage 1**: extract known axioms
  
  $\mathcal{E}$:
  
  $+(x, +(y, z)) = +(+(x, y), z)$
  $+(x, 0) = x$
  $+(x, -(x)) = 0$

  Table 1:
  
  $F(x, F(y, z)) = F(F(x, y), z) \implies \text{Ass}(F)$
  $F(x, E) = x \implies \text{Neut}_r(F, E)$
  $F(x, I(x)) = E \implies \text{Inv}_r(F, I, E)$

- **Stage 2**: match known structures on extracted axiom set

  extracted axioms:
  
  $\{\text{Ass}(+), \text{Neut}_r(+, 0), \text{Inv}_r(+, -, 0)\}$

  Table 2:
  
  $\{\text{Neut}_r(F, E), \text{Ass}(F), \text{Inv}_r(F, I, E)\}$
  $\implies \text{Group}(F, I, E)$

- Similarly staged: *theory directory* in (Kirchner, Kirchner 1994–)
Stage 2: match known structures on extracted axiom set

extracted axioms:
\{\text{Ass}(+), \text{Neut}_{r}(+, 0), \text{Inv}_{r}(+, -, 0)\}

Table 2:
\{\text{Neut}_{r}(F, E), \text{Ass}(F), \text{Inv}_{r}(F, I, E)\} \implies \text{Group}(F, I, E)

Stage 3: instantiate strategy

detected axiomatization:
\text{Group}(+, -, 0)

Table 3:
\text{Group}(F, I, E) \implies \quad >:= \text{LPO}(I>F>E), \varphi := \text{gtweight}

Start proof search with reduction ordering \text{LPO}(->+>+0) and weighting function \text{gtweight}
Ⅲ Controlling Redundancy
Efficiency of completion depends on number of rules and critical pairs generated: *Prune the search space!*

**Simplification** and *redundancy elimination*: *Safely cut off* possibly infinite bands of derivable facts
Occasionally completion finite, then word problem decidable

- Particular interest in techniques *beyond* comparing normal forms
  In the spirit of *critical pair criteria* like
  – connectedness (Winkler, Buchberger 1983)
  – compositeness (Kapur, Musser, Narendran 1985)

- Revisit redundancy criteria realized in **WALDMEISTER**
Caveat: not every criterion speeds up proof search!
Even if so: mind *trade-off* between cost and benefit

Working horse: an equation \(s = t\) *redundant* wrt. \(E\)
if every *ground instance* has a smaller proof in \(E\)
(since ordered completion only strives for *ground* confluence)

Different ground instances may enjoy *different* proofs.
Hence often *stronger* than comparing normal forms

Approach here: establish *ground joinability* \(s\sigma \downarrow_E t\sigma\)
Then proof complexity dominated by *first step* on greater side
Need only compare say \(s\sigma \rightarrow_P^u \Rightarrow_v s'\) and \(s\sigma \rightarrow_{s \Rightarrow t}^\lambda t\sigma\)
Many presentations *confluent* only on the *ground* level, e.g. for:
- AC, ACI, Boolean rings (Martin, Nipkow 1990)
- Abelian groups, rings (WM)

Improvements in presence of AC axioms *pressing*:
From these alone, *infinite* band of equations . . .
Grows $1, 3, 11, 53, 313, \ldots = \frac{1}{2}(I(n-1) + (n-1)(n-1)!) \in O(n!)$

As reduction ordering, fix an arbitrary KBO or LPO
Then $\text{ACC'} = \text{AC} \cup \{x + (y + z) = y + (x + z)\}$ *ground confluent*

*Thm.*: Every *AC-valid* $s =_m t$ outside ACC’ *redundant*
Many presentations confluent only on the ground level, e.g. for:

– AC, ACI, Boolean rings (Martin, Nipkow 1990)
– Abelian groups, rings (WM)

Improvements in presence of AC axioms pressing:

From these alone, infinite band of equations grows:

\[
\begin{align*}
(x_1 + x_2) + x_3 &= x_1 + (x_2 + x_3) \\
(x_1 + x_2) + x_3 &= x_2 + (x_1 + x_3) \\
(x_1 + x_2) + x_3 &= x_3 + (x_1 + x_2) \\
(x_1 + x_2) + x_3 &= x_3 + (x_2 + x_1) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_2 + (x_1 + (x_4 + x_3)) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_2 + (x_4 + (x_1 + x_3)) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_3 + (x_1 + (x_2 + x_4)) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_3 + (x_2 + (x_1 + x_4)) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_3 + (x_2 + (x_4 + x_1)) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_3 + (x_4 + (x_1 + x_2)) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_4 + (x_1 + (x_2 + x_3)) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_4 + (x_1 + (x_3 + x_2)) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_4 + (x_2 + (x_3 + x_1)) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_4 + (x_3 + (x_1 + x_2)) \\
(x_1 + (x_2 + (x_3 + x_4))) &= x_4 + (x_3 + (x_2 + x_1)) \\
\cdots
\end{align*}
\]

As reduction ordering, fix an arbitrary KBO or LPO.

Then $\text{ACC}' = \text{AC} \cup \{x + (y + z)\}$.

Thm.: Every AC-valid $s =_{\text{AC}} t$. 

Ground Convergent Subsystems (1)
Many presentations *confluent* only on the *ground* level, e.g. for:
- AC, ACI, Boolean rings (Martin, Nipkow 1990)
- Abelian groups, rings (WM)

Improvements in presence of AC axioms *pressing*:
From these alone, *infinite* band of equations
Grows $1, 3, 11, 53, 313, \ldots = \frac{1}{2}(I(n - 1) + (n - 1)(n - 1)!)$ $\in O(n!)$

As reduction ordering, fix an arbitrary KBO or LPO
Then $ACC' = AC \cup \{x + (y + z) = y + (x + z)\}$ *ground confluent*

*Thm.*: Every *AC-valid* $s =_m t$ outside ACC’ *redundant*
Ground Convergent Subsystems (2)

- Proof steps:
  - $s\sigma \downarrow_{ACC'} t\sigma$ **only by skeleton rewrites**, by ground confluence
  - applies in particular to crucial first step $s\sigma[u\varrho] \rightarrow_{u \Rightarrow n} s\sigma[v\varrho]$
  - complexities: $(\{s\sigma\}, s, m, t\sigma)$ **undercut** by $(\{s\sigma\}, u, n, s\sigma[v\varrho])$
    provided labels in ACC’ are minimal
  Works **the same** for ACI etc.

- Empirical finding: better **extend** ACC’ with
  $x + (y + z) = z + (x + y)$ and $x + (y + z) = z + (y + x)$

- CPs/problem | ROB005-1 | RNG027-5 | LAT023-1 | RNG035-7 | GRP180-1
---|---|---|---|---|---
WM | 305 000 | 418 000 | 130 000 | 237 000 | 83 000
WM-AC | 33 000 | 49 000 | 66 000 | 161 000 | 88 000
Ground Convergent Subsystems (2)

- Proof steps:
  - $s\sigma \downarrow_{\text{ACC'}} t\sigma$ only by skeleton rewrites, by ground confluence
  - applies in particular to crucial first step $s\sigma[u\rho] \rightarrow_{u\Rightarrow v} s\sigma[v\rho]
  - complexities: $(\{s\sigma\}, s, m, t\sigma)$ undercut by $(\{s\sigma\}, u, n, s\sigma[v\rho])$
    provided labels in ACC’ are minimal
  - Works the same for ACI etc.

- Empirical finding: better extend ACC’ with
  $x + (y + z) = z + (x + y)$ and $x + (y + z) = z + (y + x)$

- Proof problems with AC operators become feasible
  Low-budget technology: easy to implement
  (High budget: completion modulo AC (Lankford, Ballantyne 1977; Peterson, Stickel 1981; ...))
Case Analysis by Variables (1)

- Approximate ground joinability by *case split* on ordering relationships between variables (Martin, Nipkow 1990)

- **Implementation simple**: map variables to constants
  - LPO: ordering relationships mirrored in precedence
  - KBO: plus restriction on number of constants’ occurrences
  - Then run through case and check $\succ_{\text{enc}}$ in first step

- Number of cases necessary for $n$ variables:
  
  $1, 3, 13, 75, 541, \ldots = \sum_{k=1}^{n} \binom{n}{k-1} 2^{k-1} \in O(n!)$

  *Escalation*: split only on subset of variables
  - Last resort: abort at some limit
Case Analysis by Variables (2)

- *Experimental* finding: proof search often *blurred!*
  However *beneficial* if redundant equations kept *for rewriting*, but not for critical pairing: all descendants *redundant*

- **CPs/problem**
<table>
<thead>
<tr>
<th>ROB005-1</th>
<th>RNG027-5</th>
<th>LAT023-1</th>
<th>RNG035-7</th>
<th>GRP180-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>WM</td>
<td>305 000</td>
<td>418 000</td>
<td>130 000</td>
<td>237 000</td>
</tr>
<tr>
<td>WM-AC</td>
<td>33 000</td>
<td>49 000</td>
<td>66 000</td>
<td>161 000</td>
</tr>
<tr>
<td>WM-AC-GJ</td>
<td>18 000</td>
<td>54 000</td>
<td>54 000</td>
<td>148 000</td>
</tr>
</tbody>
</table>

- Criterion *not limited* to fixed theories, but most useful for AC
  Ground convergent systems for *Abelian groups* and *rings*
Confluence Trees

- **Decision procedure** for ground confluence if $>$ is LPO (Comon, Narendran, Nieuwenhuis, Rusinowitch 1998)
  LPO constraint solver of (Nieuwenhuis, Rivero 2002)

- Tree nodes marked with equation and ordering constraint
  Branching wrt. *arbitrary terms* if ordered rewriting (im)possible
  **Ground joinability** if all leaves tautologies, **redundancy** if $>_{\text{enc}}$

- Computationally **expensive**: constraint solving NP-hard already
  Trees **not unique**: one may fail, another succeed
  Implementation effort **tremendous**

- t/s [PIII 1GHz]  BOO023-1 BOO026-1 GRP181-3 RNG028-5 ROB006-1

<table>
<thead>
<tr>
<th></th>
<th>WM-GJ</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WM-GJ</td>
<td></td>
<td>&gt; 600</td>
<td>2.7</td>
<td>127.8</td>
<td>13.9</td>
</tr>
<tr>
<td>WM-CT</td>
<td>5.9</td>
<td>144.2</td>
<td>92.9</td>
<td>68.7</td>
<td>35.0</td>
</tr>
</tbody>
</table>

Th. Hillenbrand

FAST EQUATIONAL REASONING – p.36
Confluence Trees

- **Decision procedure** for ground confluence if > is LPO (Comon, Narendran, Nieuwenhuis, Rusinowitch 1998)
  LPO *constraint solver* of (Nieuwenhuis, Rivero 2002)

- Tree nodes marked with equation and ordering constraint
  Branching wrt. *arbitrary terms* if ordered rewriting (im)possible
  *Ground joinability* if all leaves tautologies, *redundancy* if $>_{\text{enc}}$

- Computationally *expensive*: constraint solving NP-hard already
  Trees *not unique*: one may fail, another succeed
  Implementation effort *tremendous*

- Effect on proof search: rather *mixed*
  May help on *individual* problems
AC Ground Reducibility

- **Aim:** *stronger* criterion for **AC case** without computational effort of confluence trees
  Idea: from AC class of $s = t$ distill *subset* w/o redundancy

- Check (permutations of) $s$ and $t$ for *ground reducibility* wrt. CC’
  Restricted to skeleton: expressible by usual *ordering constraints*

- **Necessary** criterion for constraint satisfiability, *polynomial* cost
  Closes constraint under some ordering-specific consequences

- t/h [PIII 1GHz]  ROB020-1 ROB007-1 LAT018-1 RNG036-7

<table>
<thead>
<tr>
<th></th>
<th>WM-GJ</th>
<th>WM-GR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6.0</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td>39.4</td>
<td>13.4</td>
</tr>
<tr>
<td></td>
<td>&gt; 300</td>
<td>13.2</td>
</tr>
<tr>
<td></td>
<td>888.2</td>
<td>291.2</td>
</tr>
</tbody>
</table>
Epilogue: AC Deletion Proliferated

- **Superposition** provers E (Schulz 2001) and PROVER9 (McCune 2008): *Discard* $C \lor s = t$ outside ACC’ if $\text{AC} \models s = t$

- **No correctness proof** so far – **impossible** the standard way say of (Nieuwenhuis, Rubio 2001 HAR): $\vdash$ as $\text{LPO}(+\!>\!a\!>\!b\!>\!c)$

  $\text{ACC'} \models a + (c + b) = c + (b + a)$ needs at least $a + (c + b) = c + (a + b)$

  but $\{a + (c + b), c + (b + a)\} < \{a + (c + b), c + (a + b)\}$

  Hence *not redundant*, incompleteness possible

- Remedy: **refine** definition of literal complexity. For $s\sigma > t\sigma$:

  $$(s \Join_m t)\sigma \longmapsto (\{s\sigma\}, \Join, s, m, t\sigma)$$

  Now superposition redundancy *subsumes* completion redundancy!

  Cf. framework of *canonical inference* (Dershowitz, Kirchner 2006)
IV Applications
WALDMEISTER in Practice

● Foremost: *educational*, reference implementation …

● User-reported *application areas*:
  – reasoning in specific algebraic structures
  – program transformation
  – modelling of agent systems
  – hardware verification
  – knowledge representation
  – protocol synthesis
  – disambiguation in language processing
  – modelling of bible interpretations

● Integration into *interactive systems*:
  ILF – ΩMEGA – THEOREMA – MATHEMATICA
- Foremost: educational,
- User-reported applications:
  - reasoning in specific algebraic structures
  - program transformation
  - modelling of agent systems
  - hardware verification
  - knowledge representation
  - protocol synthesis
  - disambiguation in language processing
  - modelling of bible interpretations
- Integration into interactive systems: ILF, ΩMEGA, MEGA, MATHMATICA, THEOREMA, etc.
WALDMEISTER in Practice

- Foremost: *educational*, reference implementation

- User-reported *application areas*:
  - reasoning in specific algebraic structures
  - program transformation
  - modelling of agent systems
  - hardware verification
  - knowledge representation
  - protocol synthesis
  - disambiguation in language processing
  - modelling of bible interpretations

- Integration into *interactive systems*:
  ILF – ΩMEGA – THEOREMA – MATHEMATICA
Commuting Group Endomorphisms

- **Small conflict clauses** for theory reasoners in equality with UIF

  Algebra of equality proofs (Stump, Tan 2005 RTA) \(\cong\) free groups

  *Proof mining*: canonical forms hint at *minimal* assumptions

- Adding \(k\) congruence proof rules gives theory \(\text{CGE}_k\)

  \(\text{WALDMEISTER}\) delivers \(k\) \(2\) \(3\) \(4\) \(5\)

  *ground* convergent size \(24\) \(70\) \(566\) \(11910\)

  system for small \(k\):

  \(\text{CPs}\) \(320\) \(2676\) \(229371\) \(118887623\)

- Normal forms *difficult* to characterize. But for \(k=2\):

  With \(\text{APROVE}\)-ordering system *orientable* and *convergent*

  Leads to: *generic* description (Stump, Löchner 2006),

  completion with termination checking (Slothrop 2006 RTA)
Quasigroup Problems for Theorem Provers

- (Phillips, Stanovský 2008) at upcoming ESARM workshop: Automated reasoning tools of *increasing impact* on *loop theory!* Survey *LT contributions* obtained with AR support

- Selection of 80 *representative* proof problems (*QPTP*)
  Compare performance of various automated theorem provers
  Finding: on equational problems *WALDMEISTER* performs *best*

- *Example:* Is every F-quasigroup isotopic to a Moufang loop?
  “... the result in [KKP07] was originally derived as a series of results, a number of steps eventually leading to the main theorem... Waldmeister proved it from scratch in 40 minutes.”

Had been open since 1967. [KKP07]: 27 pages in J Alg
Single Axioms for the Sheffer Stroke

- (Wolfram 2002): empirical and systematic study of *computational systems* such as cellular automata, Turing machines, *operator systems*
  In every class, among *simplest* cases always instances of *great* complexity

- *Simplest* axiomatizations of *Boolean algebra?*
  
  Thm.: 
  \[(x \mid y) \mid z) \mid (x \mid ((x \mid z) \mid x)) = z\]
  specifies *Sheffer stroke*
  Proved with *WALDMEISTER* and reprinted . . .
Single Axioms for the Sheer Stroke

(Wolfram 2002): empirical and systematic study of computational systems such as cellular automata, Turing machines, operator systems. In every class, among simplest cases always instances of great complexity.

Simplest axiomatizations of Boolean algebra:

Thm.: 

\[ (x \rightarrow y) \rightarrow ((x \rightarrow z) \rightarrow (x \rightarrow (x \rightarrow z))) = z \]

specifies Sheffer stroke

Proved with WALDMEISTER and reprinted

Th. Hillenbrand FAST EQUATIONAL REASONING { p.43
Single Axioms for the Sheffer Stroke

- (Wolfram 2002): empirical and systematic study of *computational systems* such as cellular automata, Turing machines, *operator systems*. In every class, among *simplest* cases always instances of *great* complexity.

- Recognizes *progress in AR* over the decades:
  “Ever since the 1970s I at various times investigated using automated theorem-proving systems. But it always seemed that extensive human input . . . was needed to make such systems actually find non-trivial proofs. In the late 1990s, however, I decided to try the latest systems and was surprised that some of them could routinely produce proofs hundreds of steps long with little or no guidance.”
Integration into MATHEMATICA

- **Consequence** of these experiments:
  
  “We are interested in adding theorem proving capabilities to MATHEMATICA.” (Oct. 2002)

- Introduced SW engineers of Wolfram, Inc. into WM code
  System had to become *re-entrant*, danger of *memory leaks*
  Patent attorneys of MPG worked out *license agreement*

- Functionality *available* since release of version 6.0 in mid-2007
  Encapsulated within `FullSimplify[expr, assum]` ...
Integration into Mathematica

- **Consequence** of these experiments:
  
  “We are interested in adding theorem proving capabilities to Mathematica.” (Oct. 2002)

- Introduced SW engineers of Wolfram, Inc. into WM code
  
  System had to become re-entrant, danger of memory leaks
  
  Patent attorneys of MPG worked out license agreement

- Functionality **available** since release of version 6.0 in mid-2007
  
  Encapsulated within `FullSimplify`

**Equational Theorem Proving**

Mathematica 6 for the first time brings general automated theorem proving into an immediate interactive environment. Extending Mathematica’s already uniquely powerful algebraic theorem-proving capabilities,

Mathematica 6 introduces equational theorem proving capable of operating on industrial-scale arbitrary abstract systems of axioms or relations, and integrating theorem proving into the computational workflow.

- Advanced equational theorem proving automatically accessed directly from `FullSimplify`.
- Full support for `ForAll`, `Exists`, etc. quantifiers.
- Immediately allows Mathematica arbitrary symbolic notation for maximum readability.
- Uses state-of-the-art unfailing completion methods.
• **Consequence** of these experiments:

  “We are interested in adding theorem proving capabilities to MATHEMATICA.” (Oct. 2002)

• Introduced SW engineers of Wolfram, Inc. into WM code

  System had to become re-entrant, danger of memory leaks

  Patent attorneys of MPG worked out license agreement

• Functionality **available** since release of version 6.0 in mid-2007

  Encapsulated within `FullSimplify`
Integration into MATHEMATICA

- **Consequence** of these experiments:
  “We are interested in adding theorem proving capabilities to MATHEMATICA.” (Oct. 2002)

- Introduced SW engineers of Wolfram, Inc. into WM code
  System had to become *re-entrant*, danger of *memory leaks*
  Patent attorneys of MPG worked out *license agreement*

- Functionality *available* since release of version 6.0 in mid-2007
  Encapsulated within `FullSimplify[expr, assum]`

- Gives *evidence* that automated theorem proving is spreading
  Seize the opportunity!
Conclusion

- Analysis of *proof procedure* leads to smart system design

- *Prover engineering* produces high-performance system

- *Controlling redundancy* is the key to solving difficult problems

- Taking all this together, *applications* are out there somewhere

- *Future work* includes:
  - Horn theories, by the lazy programmer
  - joint efforts on open problems
References (1)


Graf 1996: Term Indexing. LNCS 1053.

Dershowitz, Kirchner 2006: Abstract canonical presentations. TCS 357(1–3).

Huet 1981: A complete proof of correctness of the Knuth-Bendix completion algorithm. JCSS 23.

Kapur, Musser, Narendran 1985: Only prime superpositions need be considered in the Knuth-Bendix procedure. GE Report.
Kirchner, Kirchner 1994–: Rewriting Solving Proving. See authors’ web pages.
Lankford 1975: Canonical inference. ATP-32, UT Austin.
Martin, Nipkow 1990: Ordered rewriting and confluence. CADE-10.


References (4)

Stump, Löchner 2006: Knuth-Bendix completion of theories of commuting group endomorphisms. IPL 98(5).


Schulz 2001: System abstract: E 0.61. IJCAR-1.

Winkler, Buchberger 1983: A criterion for eliminating unnecessary reductions in the Knuth-Bendix algorithm. CACLCS.


Wos 1992: Note on McCune’s article on discrimination trees. JAR 9(2).

Wos, Carson, Robinson 1964: The unit preference strategy in theorem proving. AFIPSP 26(1).