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):

	WM	VAMPIRE	E	OTTER	METIS	EQUINOX	GEO
solved	91	63	59	27	15	2	2
av. time	18.2	42.3	16.7	21.6	38.3	13.4	255.8

- What are the *underlying concepts?*

Outline

- Foundations
- Prover engineering
- Controlling redundancy
- Applications

I Foundations

Equational Logic

- *Example:* group axiomatization

$$E: \ (x+y)+z = x+(y+z) \qquad x+0=x \qquad x+(-x)=0$$

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

$$s \longleftrightarrow_{u \Rightarrow_m v} t \quad \longmapsto \quad (\{s\}, u, m, t) \quad \text{if } s \succ t$$

- Deduction of new facts must ensure *fairness:* eventually smaller proof for every persistent *ground peak* $s \longleftarrow t \longrightarrow u$ in Σ^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 . . .

WALDMEISTER Searching for a Proof

```
**********************************************************************
************************* COMPLETION - PROOF *************************
**********************************************************************

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,+(-(x1),x2)) -> +(0,x2)
new rule:                    7  +(0,-(-(x1))) -> x1
new rule:                    8  +(x1,-(-(x2))) -> +(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
        +---------------------------+
        |   this proves the goal    |
        +---------------------------+

Proved Goals:
No.  1:  c ?= -(-(c)) joined, current: c = c
1 goal was specified, which was proved.

Waldmeister states: Goal proved.
```

WALDMEISTER Presenting a Proof

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: $0 + (- - x) = x$

$0 + (- - x)$
= by Axiom 2 RL
$(x + (-x)) + (- - x)$
= by Axiom 3 LR
$x + ((-x) + (- - x))$
= by Axiom 2 LR
$x + 0$
= by Axiom 1 LR
x

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

$x + (- - y)$
= by Axiom 1 RL
$(x + 0) + (- - y)$
= by Axiom 3 LR
$x + (0 + (- - y))$
= by Lemma 1 LR
$x + y$

Lemma 3: $0 + x = x$

$0 + x$
= by Lemma 2 RL
$0 + (- - x)$
= by Lemma 1 LR
x

Theorem 1: $x = - - x$

x
= by Lemma 3 RL
$0 + x$
= by Lemma 2 RL
$0 + (- - x)$
= by Lemma 3 LR
$- - x$

Calculus and Proof Procedure

- Ordered / unfailing completion: given as set of *calculus rules*

  expanding: $\dfrac{l = r \quad s[l'] = t}{(s[r] = t)\sigma}$ critical pairing

  contracting: rewrite-based simplification rules

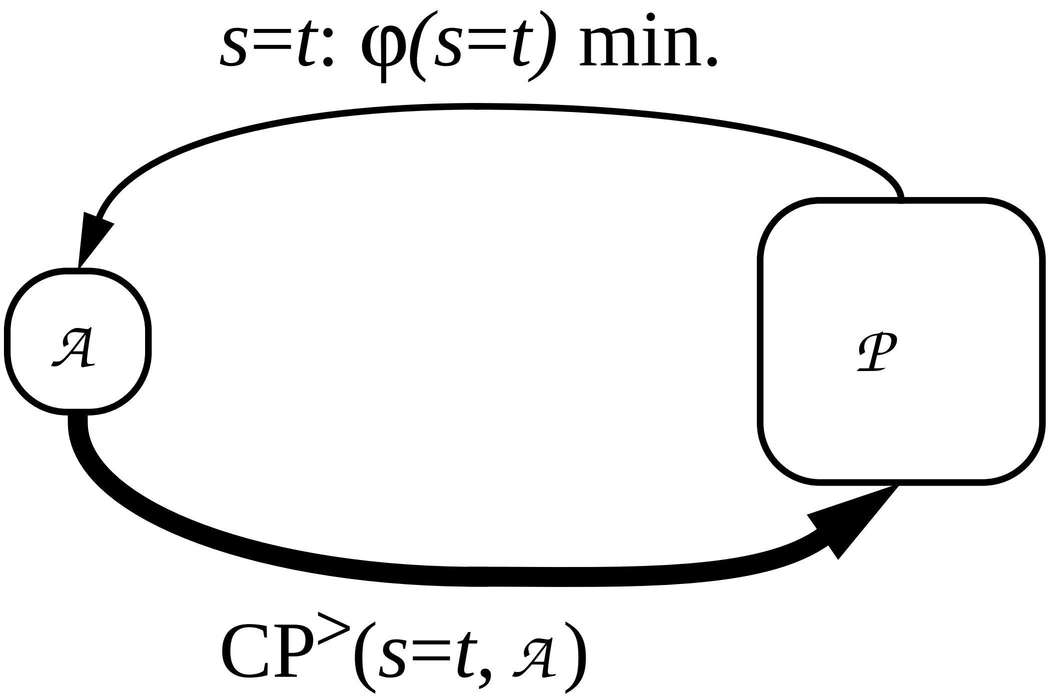
- Additional *control constraint:* fairness
  Parameter: reduction ordering

- 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 φ

- *Active* facts $\mathcal{A}$ for rewriting and superposition
  Passive facts $\mathcal{P}$: critical pairs descending from $\mathcal{A}$



Proof Procedure

FUNCTION $\textsc{Waldmeister}(\Sigma, \mathcal{E}, \mathcal{C}, >, \varphi)$: BOOL

$(\mathcal{A}, \mathcal{P}) := (\varnothing, \mathcal{E})$

WHILE $\neg\mathrm{trivial}(\mathcal{C}) \wedge \mathcal{P} \neq \varnothing$ **DO**

$\quad e := \min_\varphi(\mathcal{P});\ \mathcal{P} := \mathcal{P} \setminus \{e\}$

$\quad e := \mathrm{Normalize}^{>}_{\mathcal{A}}(e)$

$\quad$ **IF** $\neg\mathrm{redundant}(e)$ **THEN**

$\quad\quad (\mathcal{A}, P_1) := \mathrm{Interred}^{>}(\mathcal{A}, e)$

$\quad\quad \mathcal{A} := \mathcal{A} \cup \{\mathrm{Orient}^{>}(e)\}$

$\quad\quad P_2 := \mathrm{CP}^{>}(e, \mathcal{A})$

$\quad\quad \mathcal{P} := \mathrm{Update}(\mathcal{P} \cup P_1 \cup P_2)$ Normalize...

$\quad\quad \mathcal{C} := \mathrm{Normalize}^{>}_{\mathcal{A}}(\mathcal{C})$

$\quad$ **END**

END

RETURN $\mathrm{trivial}(\mathcal{C})$

Proof Procedure

FUNCTION $\textsc{Waldmeister}(\Sigma, \mathcal{E}, \mathcal{C}, >, \varphi)$: BOOL

$(\mathcal{A}, \mathcal{P}) := (\varnothing, \mathcal{E})$
WHILE $\neg\mathrm{trivial}(\mathcal{C}) \wedge \mathcal{P} \neq \varnothing$ **DO**
  $e := \min_\varphi(\mathcal{P});\ \mathcal{P} := \mathcal{P} \setminus \{e\}$
  $e := \mathrm{Normalize}^{>}_{\mathcal{A}}(e)$
  IF $\neg\mathrm{redundant}(e)$ **THEN**
    $(\mathcal{A}, P_1) := \mathrm{Interred}^{>}(\mathcal{A}, e)$
    $\mathcal{A} := \mathcal{A} \cup \{\mathrm{Orient}^{>}(e)\}$
    $P_2 := \mathrm{CP}^{>}(e, \mathcal{A})$
    $\mathcal{P} := \mathrm{Normalize}^{>}_{\mathcal{A}}(\mathcal{P} \cup P_1 \cup P_2)$ OTTER loop – eager
    $\mathcal{C} := \mathrm{Normalize}^{>}_{\mathcal{A}}(\mathcal{C})$
  END
END
RETURN $\mathrm{trivial}(\mathcal{C})$

Proof Procedure

FUNCTION $\textsc{Waldmeister}(\Sigma, \mathcal{E}, \mathcal{C}, >, \varphi)$: BOOL

$(\mathcal{A}, \mathcal{P}) := (\varnothing, \mathcal{E})$

WHILE $\neg\mathrm{trivial}(\mathcal{C}) \wedge \mathcal{P} \neq \varnothing$ **DO**

    $e := \min_{\varphi}(\mathcal{P});\ \mathcal{P} := \mathcal{P} \setminus \{e\}$

    $e := \mathrm{Normalize}^{>}_{\mathcal{A}}(e)$

    IF $\neg\mathrm{redundant}(e)$ **THEN**

        $(\mathcal{A}, P_1) := \mathrm{Interred}^{>}(\mathcal{A}, e)$

        $\mathcal{A} := \mathcal{A} \cup \{\mathrm{Orient}^{>}(e)\}$

        $P_2 := \mathrm{CP}^{>}(e, \mathcal{A})$

        $\mathcal{P} := \mathcal{P} \cup \mathrm{Normalize}^{>}_{\mathcal{A}}(P_1 \cup P_2)$ DISCOUNT loop – lazy

        $\mathcal{C} := \mathrm{Normalize}^{>}_{\mathcal{A}}(\mathcal{C})$

    END

END

RETURN $\mathrm{trivial}(\mathcal{C})$

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

- *Entitities* 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

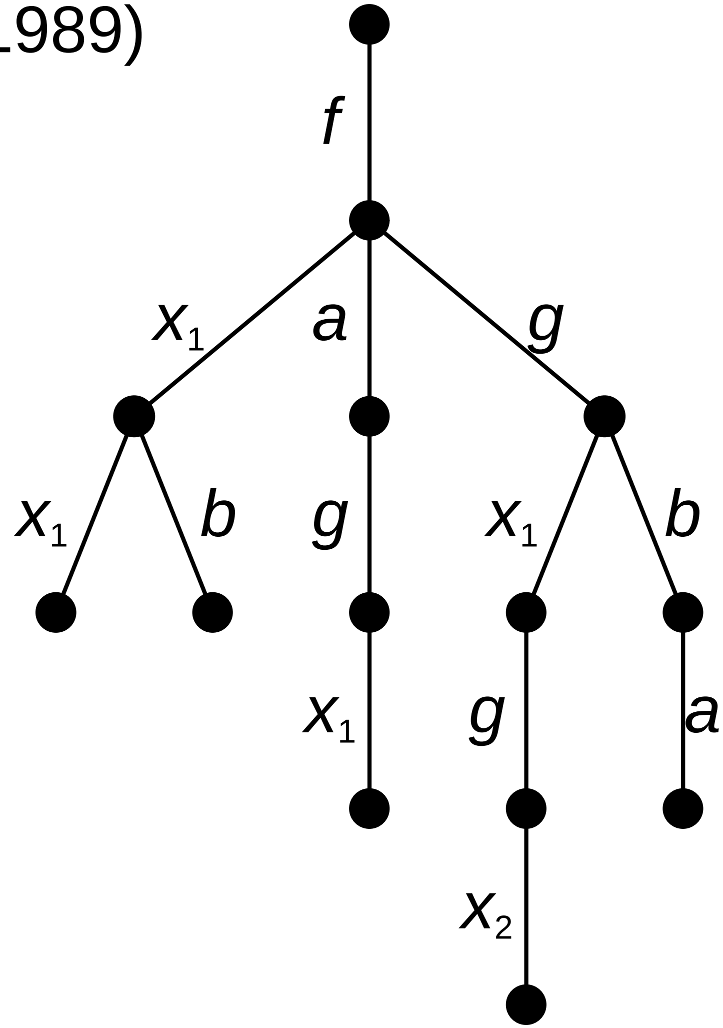
  $f(x_1, x_1)$

  $f(x_1, b)$

  $f(a, g(x_1))$

  $f(g(x_1), g(x_2))$

  $f(g(b), a)$

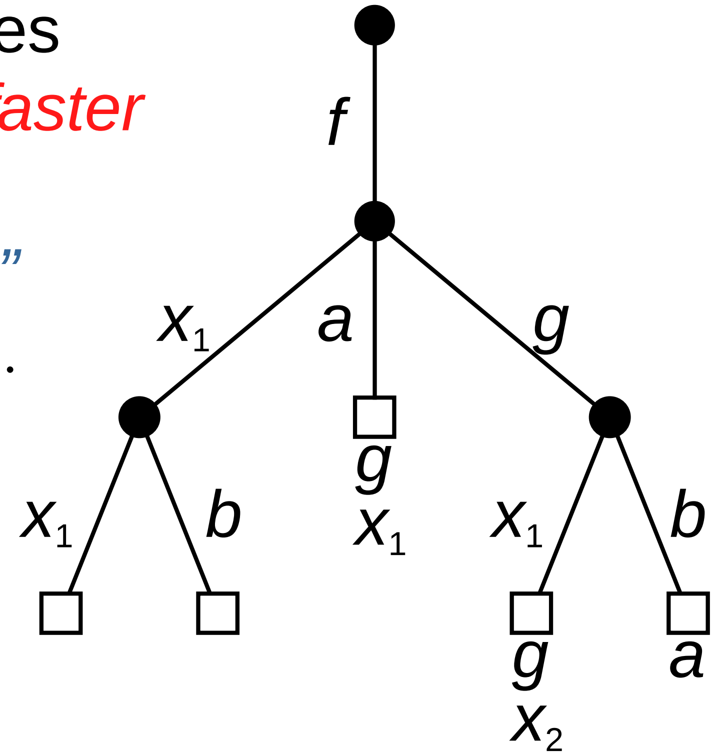


- Retrieval typically via *backtracking*
  due to *non-determinism* in descent

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: . . .

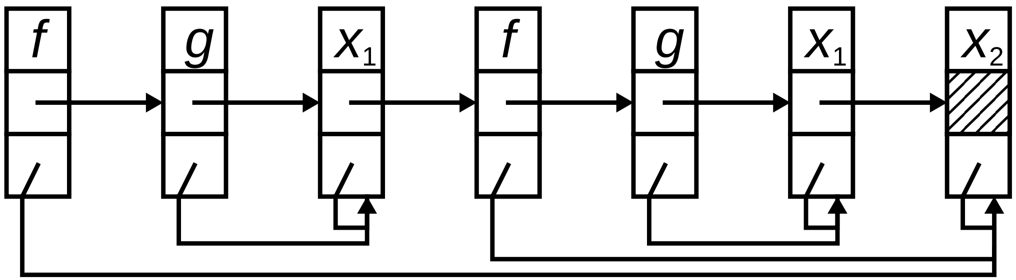


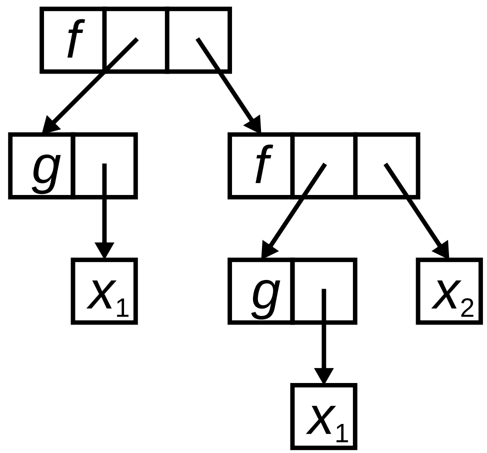
Discrimination Trees (2)

- 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	:	discr. 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 φ: 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:

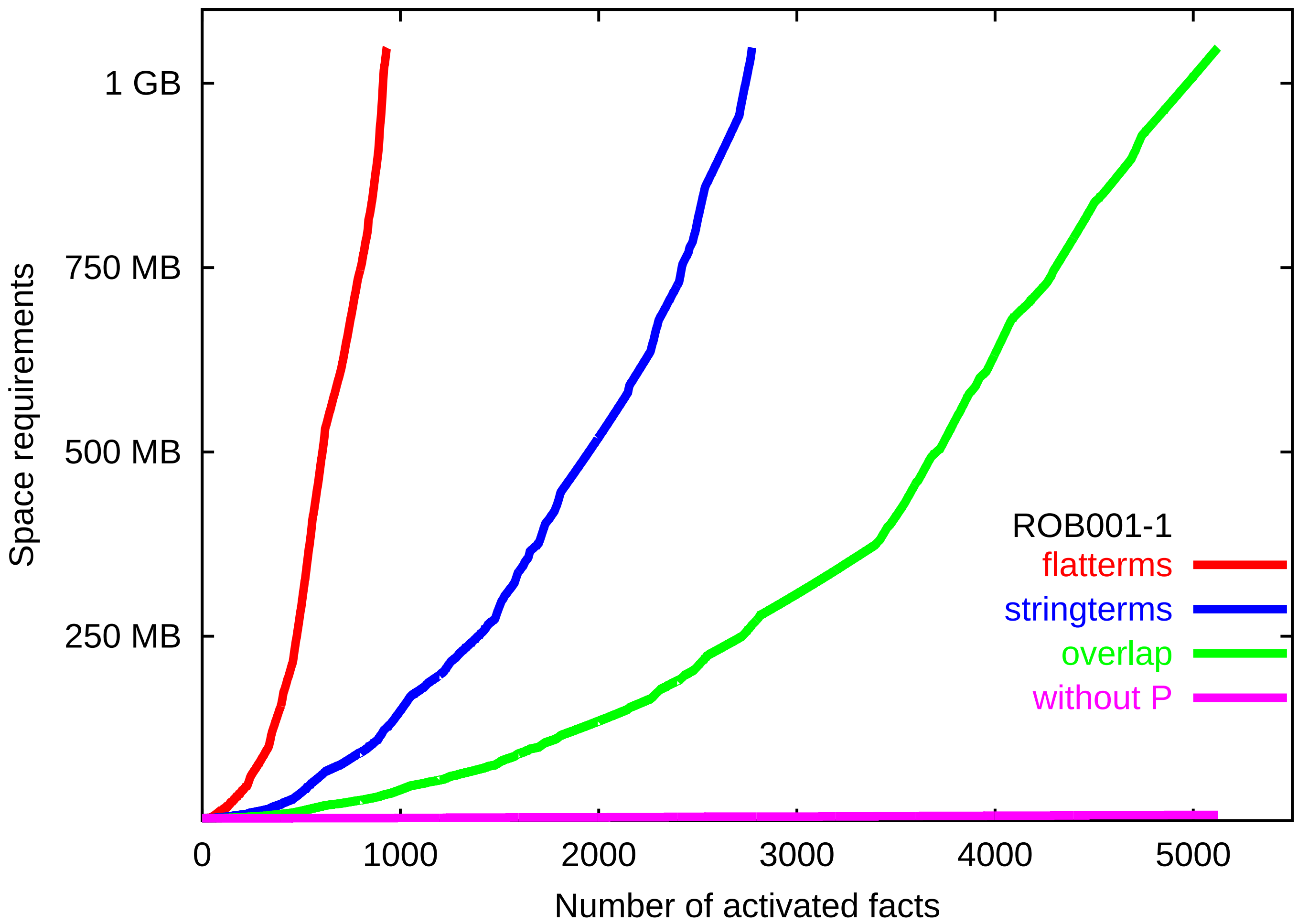
flatterms $f - \underline{x_1} - f - \underline{a} - \underline{x_2} — f - \underline{x_1} - \underline{x_2}$

stringterms

f	x_1	f	a	x_2	f	x_1	x_2

implicit $<s[l']_p=t,\ l=r>$

Space Behaviour over Time



Towards the WALDMEISTER Loop

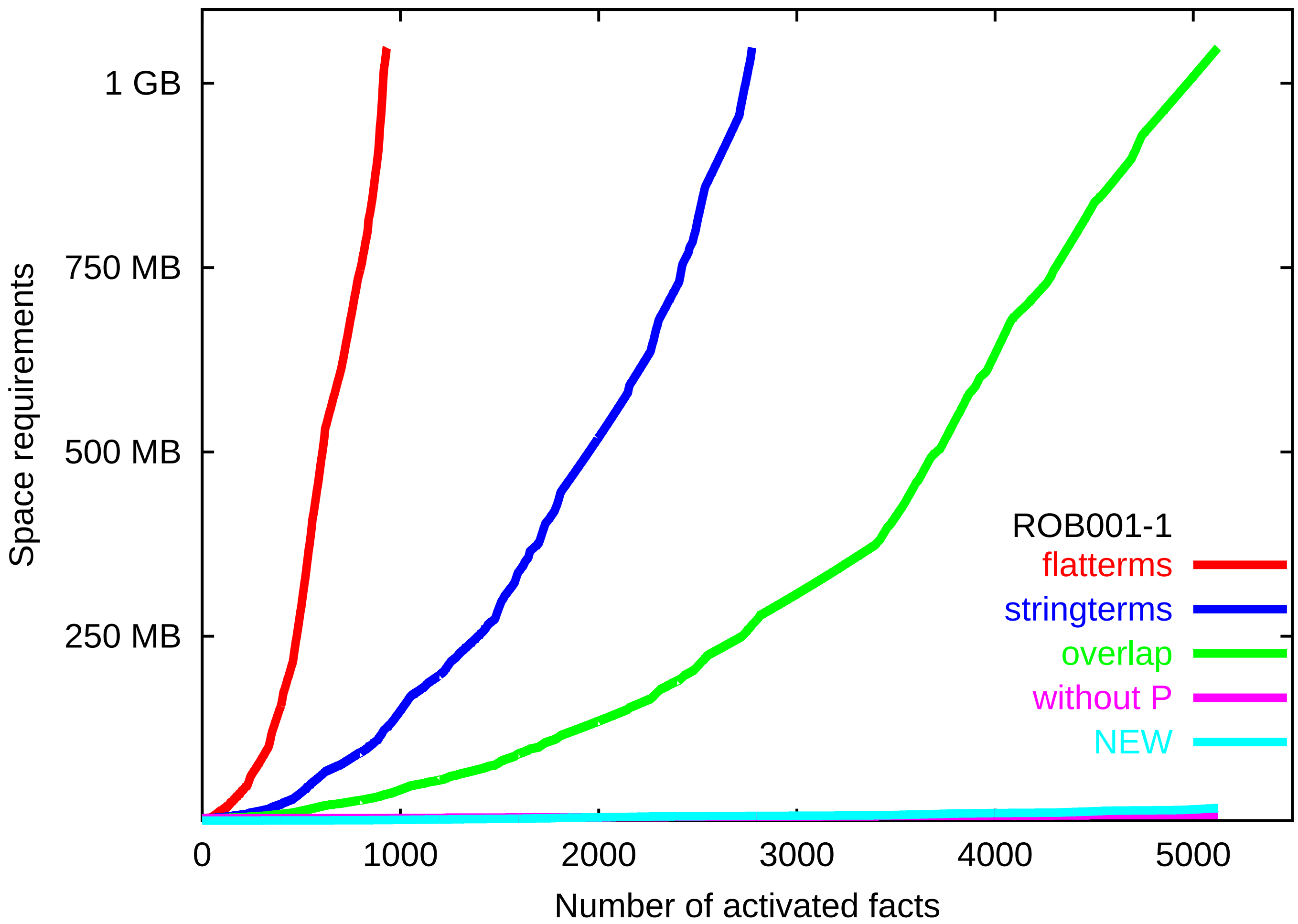
- *Group together* elements generated during *same* loop iteration: themselves *ordered* by φ, occasional removal of *lightest* element

- 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 $\mathcal{A}$ fits into *one DT* with *age constraints*
  Prerequisite for practicality: *cache* for lightweight entries

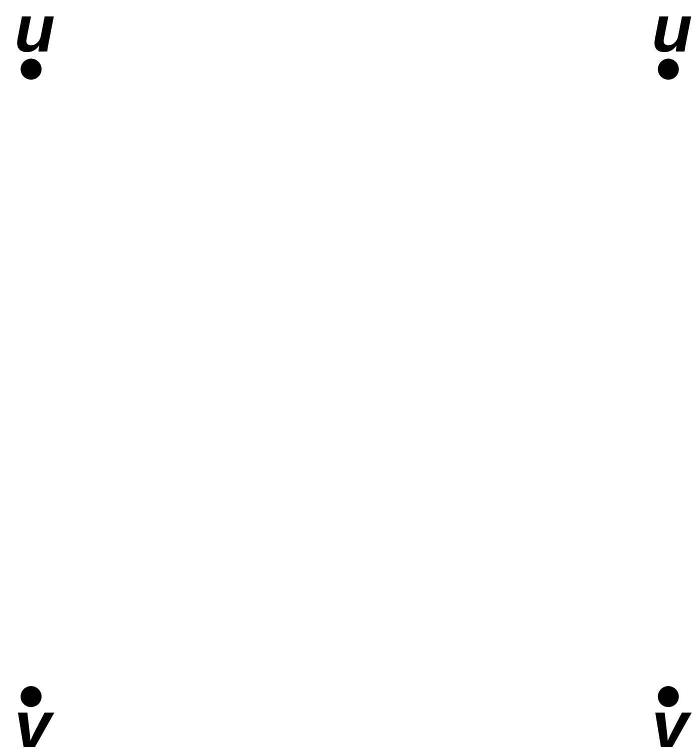
- All in all: space for $\mathcal{P}$ *linear* in $|A|$. *Laziness works!*
  Besides: *proof objects* for free, *parallelization* possible

Space Behaviour over Time (revisited)



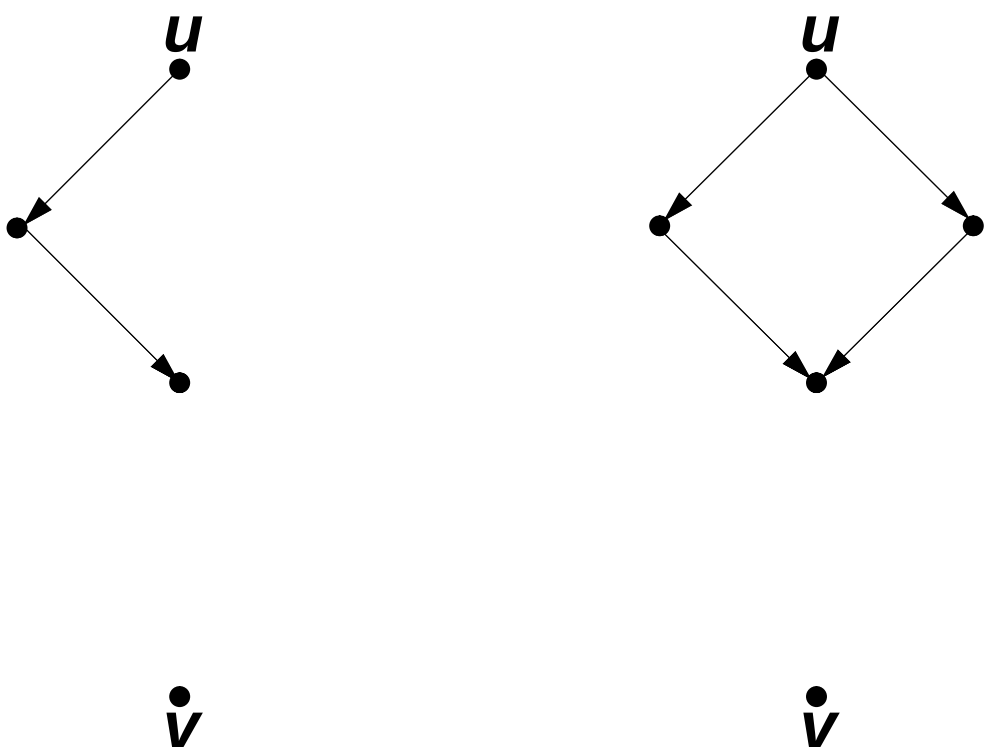
Representing the Conjecture

- 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



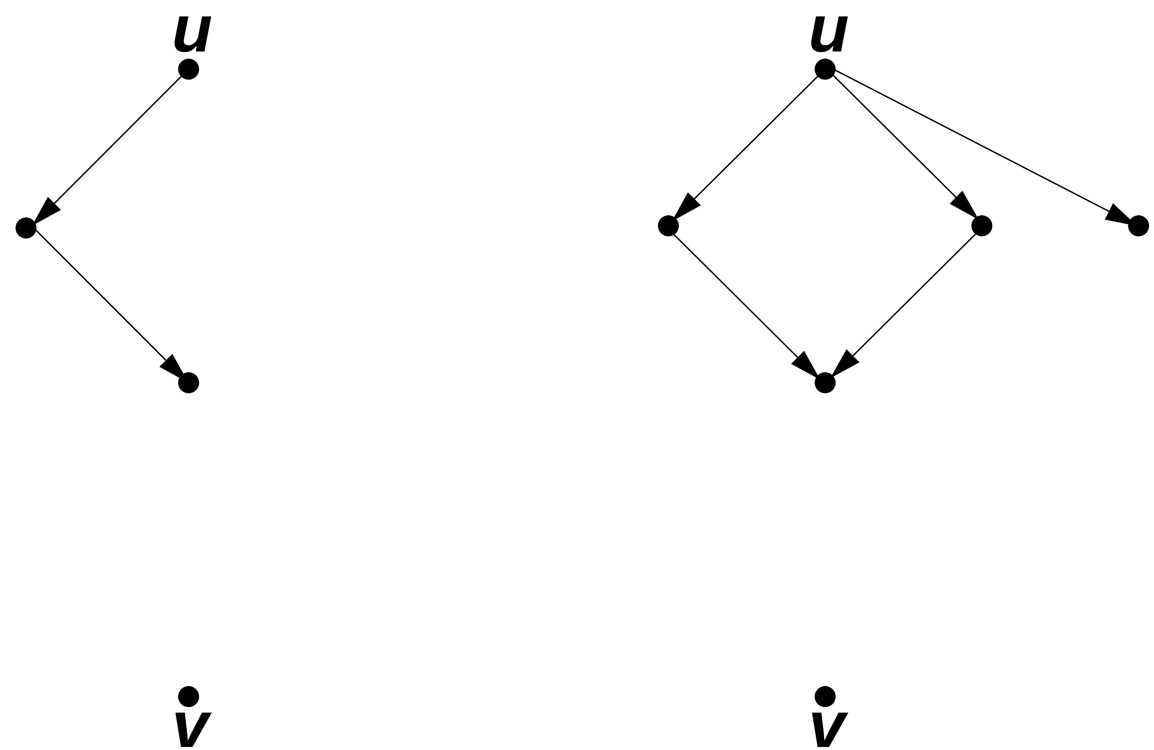
Representing the Conjecture

- 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



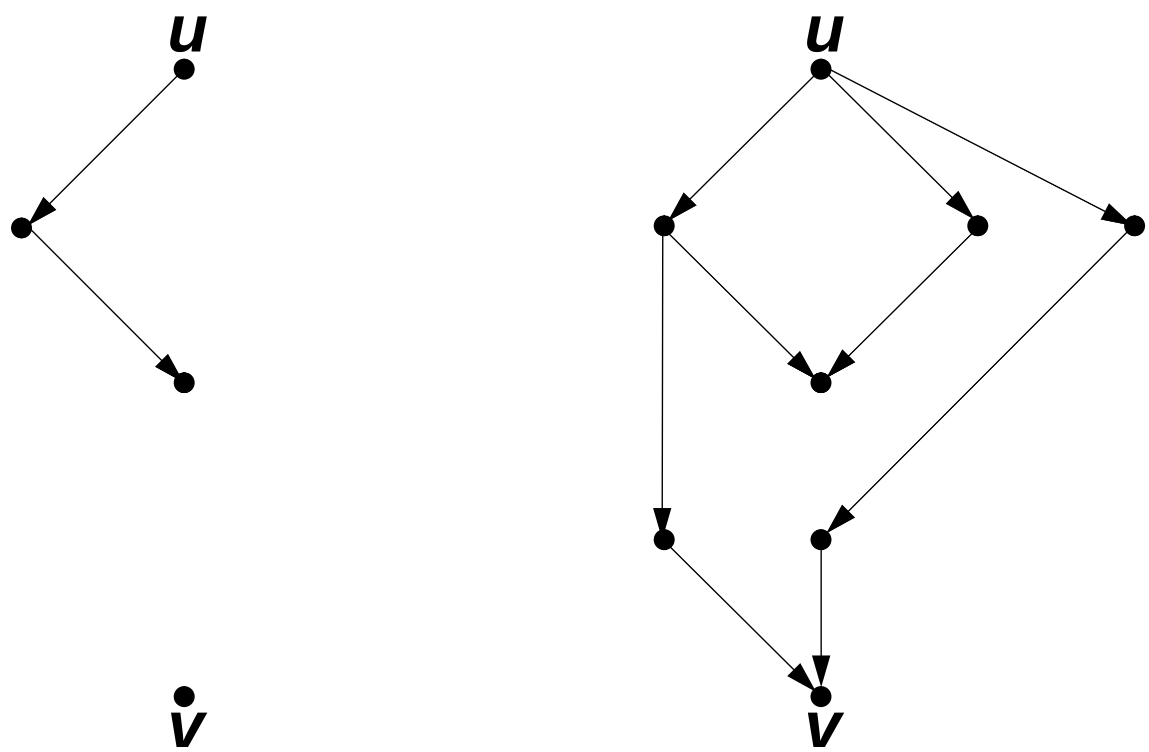
Representing the Conjecture

- 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



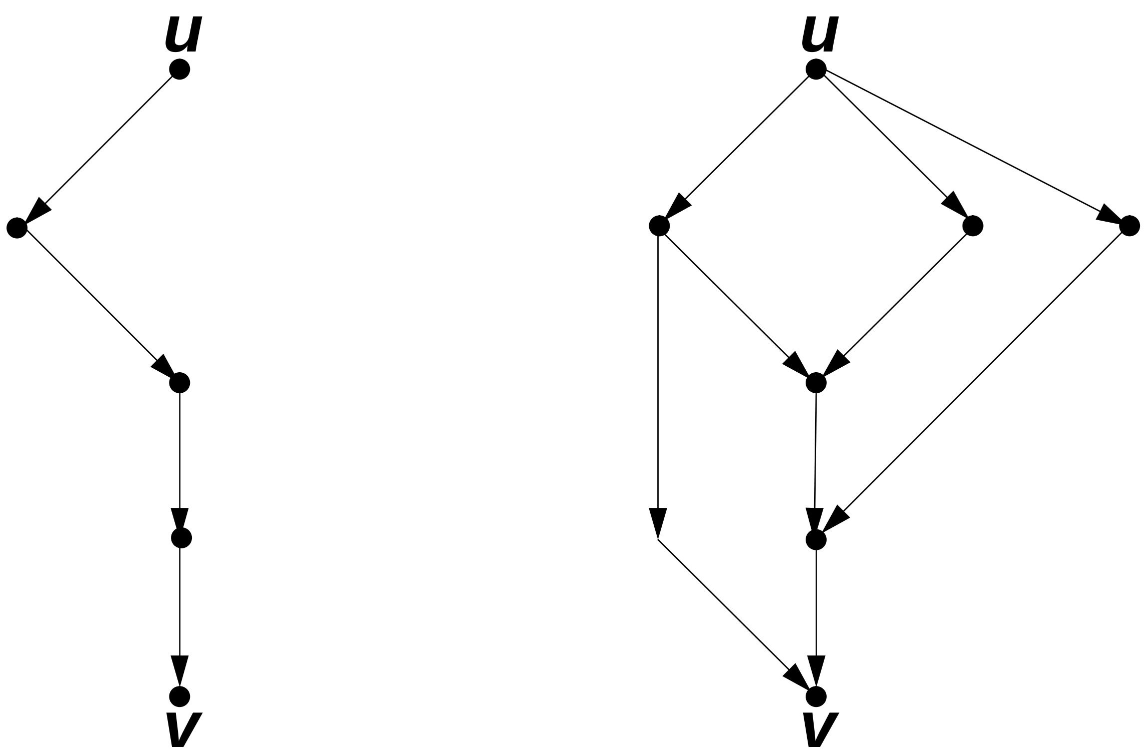
Representing the Conjecture

- 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 *termpair,* 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* φ in various domains

t/s [SPARC]	addweight	gtweight
BOO003-2	>300	0.1
BOO007-2	>300	81.8
BOO008-4	61.1	7.0
LCL153-1	2.1	>300
LCL154-1	2.0	>300
LCL155-1	1.2	>300
Σ Boolean	22 / 29	29 / 29
	25.4	4.5
Σ Wajsberg	21 / 25	17 / 25
	0.9	0.9

- 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

t/s [SPARC]	LPO	KBO
COL063-4	223.0	0.0
COL063-6	>300	0.0
COL064-6	>300	0.0
Σ BT fragment	21 / 27	25 / 27
	16.6	0.5
Σ non-associa-	21 / 38	11 / 38
tive rings	3.0	1.4
	$A > C > * > - > + > 0$	
Σ lattice-ordered	98 / 102	90 / 102
groups	12.7	23.8
	$+ > \wedge > - > \vee > 0$	

- Must employ *different orderings* on different structures!

Control Component (1)

- *Recognize* known axiomatizations within input specification $\mathcal{E}$
- *Stage 1:* extract known axioms

$\mathcal{E}$:	Table 1:
$+(x,+(y,z)) = +(+(x,y),z)$	$F(x,F(y,z)) = F(F(x,y),z) \Longrightarrow \mathrm{Ass}(F)$
$+(x,0) = x$	$F(x,E) = x \Longrightarrow \mathrm{Neut_r}(F,E)$
$+(x,-(x)) = 0$	$F(x,I(x)) = E \Longrightarrow \mathrm{Inv_r}(F,I,E)$

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

extracted axioms:	Table 2:
$\{\mathrm{Ass}(+), \mathrm{Neut_r}(+,0), \mathrm{Inv_r}(+,-,0)\}$	$\{\mathrm{Neut_r}(F,E), \mathrm{Ass}(F), \mathrm{Inv_r}(F,I,E)\}$ $\Longrightarrow \mathrm{Group}(F,I,E)$

- Similarly staged: *theory directory* in (Kirchner, Kirchner 1994–)

Control Component (2)

- Stage 2: match known structures on extracted axiom set

extracted axioms:	Table 2:
$\{\mathrm{Ass}(+), \mathrm{Neut}_r(+, 0), \mathrm{Inv}_r(+, -, 0)\}$	$\{\mathrm{Neut}_r(F, E), \mathrm{Ass}(F), \mathrm{Inv}_r(F, I, E)\}$ $\Longrightarrow \mathrm{Group}(F, I, E)$

- *Stage 3:* instantiate strategy

detected axiomatization:	Table 3:
$\mathrm{Group}(+, -, 0)$	$\mathrm{Group}(F, I, E) \Longrightarrow$ $> := \mathrm{LPO}(I>F>E), \varphi := \mathrm{gtweight}$

- *Start proof search* with reduction ordering $\mathrm{LPO}(->+>0)$ and weighting function $\mathrm{gtweight}$

III Controlling Redundancy

Introduction (1)

- Efficiency of completion depends on number of rules and critical pairs generated: *Prune the search space!*

- *Simplification* and *redundancy elimination:*
  Safely cut off possiby 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

Introduction (2)

- *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 $s\sigma \longrightarrow^{p}_{u \Rightarrow v} s'$ and $s\sigma \longrightarrow^{\lambda}_{s \Rightarrow t} t\sigma$

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 $\mathrm{ACC}' = \mathrm{AC} \cup \{x + (y + z) = y + (x + z)\}$ *ground confluent*

- *Thm.:* Every *AC-valid* $s =_m t$ outside ACC' *redundant*

Ground Convergent Subsystems (1)

- Many presentations *conflue*
  – AC, ACI, Boolean rings (N
  – Abelian groups, rings (W
- Improvements in presence
  From these alone, *infinite* b
  Grows $1, 3, 11, 53, 313, \ldots =$
- As reduction ordering, fix a
  Then $\mathrm{ACC}' = \mathrm{AC} \cup \{x + (y$
- *Thm.:* Every *AC-valid* $s =_m$

$$
\begin{aligned}
&(x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)\\
&x_1 + x_2 = x_2 + x_1\\
&x_1 + (x_2 + x_3) = x_2 + (x_1 + x_3)\\
&x_1 + (x_2 + x_3) = x_3 + (x_1 + x_2)\\
&x_1 + (x_2 + x_3) = x_3 + (x_2 + x_1)\\
&x_1 + (x_2 + (x_3 + x_4)) = x_2 + (x_1 + (x_4 + x_3))\\
&x_1 + (x_2 + (x_3 + x_4)) = x_2 + (x_4 + (x_1 + x_3))\\
&x_1 + (x_2 + (x_3 + x_4)) = x_3 + (x_1 + (x_2 + x_4))\\
&x_1 + (x_2 + (x_3 + x_4)) = x_3 + (x_2 + (x_1 + x_4))\\
&x_1 + (x_2 + (x_3 + x_4)) = x_3 + (x_2 + (x_4 + x_1))\\
&x_1 + (x_2 + (x_3 + x_4)) = x_3 + (x_4 + (x_1 + x_2))\\
&x_1 + (x_2 + (x_3 + x_4)) = x_4 + (x_1 + (x_2 + x_3))\\
&x_1 + (x_2 + (x_3 + x_4)) = x_4 + (x_1 + (x_3 + x_2))\\
&x_1 + (x_2 + (x_3 + x_4)) = x_4 + (x_2 + (x_3 + x_1))\\
&x_1 + (x_2 + (x_3 + x_4)) = x_4 + (x_3 + (x_1 + x_2))\\
&x_1 + (x_2 + (x_3 + x_4)) = x_4 + (x_3 + (x_2 + x_1))\\
&\qquad\vdots
\end{aligned}
$$

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 $\mathrm{ACC}' = \mathrm{AC} \cup \{x + (y + z) = y + (x + z)\}$ *ground confluent*

- *Thm.:* Every *AC-valid* $s =_m t$ outside ACC' *redundant*

Ground Convergent Subsystems (2)

- Proof steps:
  – $s\sigma \downarrow_{\mathrm{ACC}'} t\sigma$ *only by skeleton rewrites*, by ground confluence
  – applies in particular to crucial first step $s\sigma[u\rho] \longrightarrow_{u \Rightarrow_n v} s\sigma[v\rho]$
  – 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 $x + (y + z) = z + (y + x)$

-

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:
  – $s\sigma \downarrow_{ACC'} t\sigma$ *only by skeleton rewrites*, by ground confluence
  – applies in particular to crucial first step $s\sigma[u\rho] \longrightarrow_{u \Rightarrow_n v} s\sigma[v\rho]$
  – 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' occurences
  Then run through case and check $>_{\mathsf{enc}}$ in first step

- Number of cases necessary for n variables:
  grows $1, 3, 13, 75, 541, \ldots = \sum_{k=1}^{n} \left\langle {n \atop k-1} \right\rangle 2^{k-1} \in O(n!)$
  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	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
WM-AC-GJ	18 000	54 000	54 000	148 000	65 000

- 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 $>_{\mathsf{enc}}$
- 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
WM-GJ	> 600	2.7	127.8	13.9	44.9
WM-CT	5.9	144.2	92.9	68.7	35.0

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 $>_{\mathsf{enc}}$
- 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
WM-GJ	6.0	39.4	> 300	888.2
WM-GR	2.6	13.4	13.2	291.2

Epilogue: AC Deletion Proliferated

- *Superposition* provers E (Schulz 2001) and PROVER9 (McCune 2008): *Discard* $C \vee s = t$ outside ACC' if AC $\models s = t$

- *No correctness proof* so far – *impossible* the standard way
  say of (Nieuwenhuis, Rubio 2001 HAR): $>$ as LPO$(+ > a > b > c)$
  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 $s\sigma > t\sigma$:

$$(s \bowtie_m t)\sigma \longmapsto (\{s\sigma\}, \bowtie, s, m, t\sigma)$$

  Now superposition redundancy *subsumes* completion redundancy!
  Cf. framework of *canonical inference* (Dershowitz, Kirchner 2006)

IV Applications

WALDMEISTER in Practice

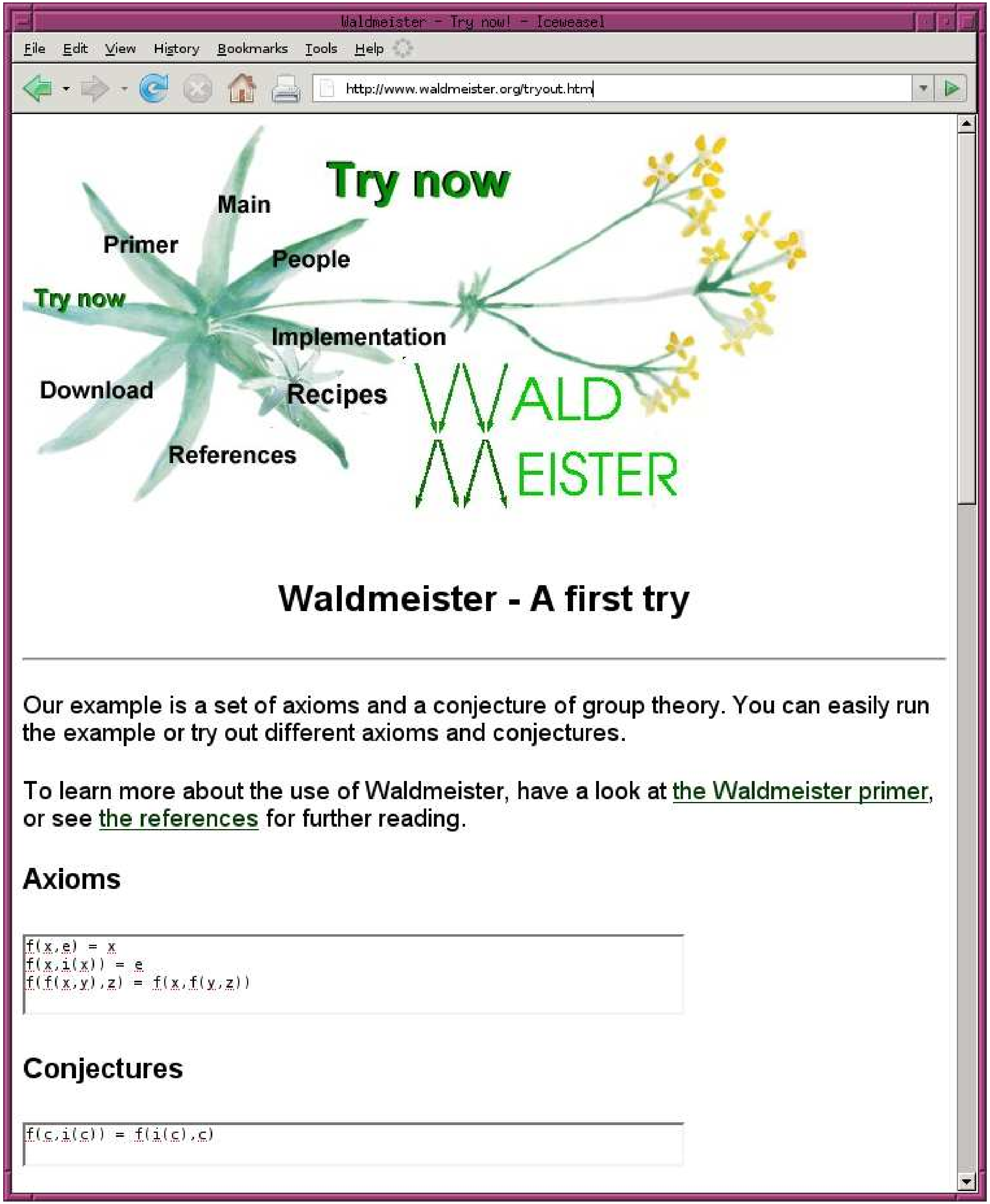
- Foremost: *educational,* reference implementation ...

- 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

WALDME

- Foremost: *educational,*
- User-reported *applicati*
  - reasoning in specific
  - program transformati
  - modelling of agent sy
  - hardware verification
  - knowledge represent
  - protocol synthesis
  - disambiguation in lan
  - modelling of bible inte
- Integration into *interact*
  ILF – ΩMEGA –



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 CGE_k

WALDMEISTER delivers	k	2	3	4	5
ground convergent	size	24	70	566	11910
system for small k:	CPs	320	2676	229371	118887623

- Normal forms *difficult* to characterize. But for k=2:
  With 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*

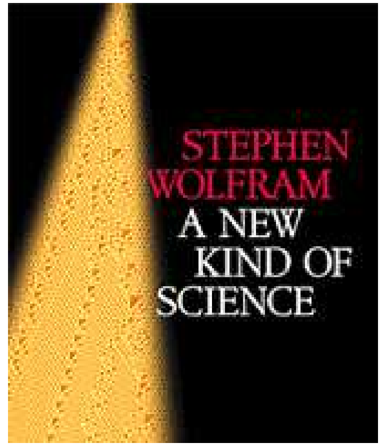
- *Example:* Is every F-quasigroup isotopic to a Moufang loop?

  > "... 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 ...

Sing

- (Wolfram
of *compu*
automata
In every c
instances

- *Simplest*
Thm.: ((x
Proved wi

= L42 ((ab)(ca))((ab)((ca)b))
= L44 ((ab)(ca))b

L49 b((ab)(ca))
= L42 ((ab)(ca))b
= L48 ab

L50 (ab)c
= L43 (a((ab)c))((ac)((ab)c))
= L43 (a((ab)c))c
= L42 c(a((ab)c))

L51 a(b(ab))
= L42 a((ab)b)
= L42 ((ab)b)a
= L50 a((ab)(((ab)b)a))
= L42 a((ba)(((ab)b)a))
= L44 aa

L52 (ba)(ab)
= L50 (ab)(b((ba)(ab)))
= L47 (ab)(ab)

L53 (aa)((ba)(ba))
= L44 (aa)((ba)(((aa)(ba))(((aa)(ba))(aa))(ba))))
= L50 (aa)((((aa)(ba))(aa))(ba))
= L42 (aa)((ba)(((aa)(ba))(aa)))
= L42 (aa)((ba)((aa)((aa)(ba))))
= L40 (aa)((ba)((aa)a))
= L42 (aa)((ba)(a(aa)))
= L47 a(aa)

L54 ((ab)(ab))((ab)(ab))
= L52 ((ba)(ab))((ba)(ab))
= L52 ((ab)(ba))((ba)(ab))
= L52 ((ab)(ba))((ab)(ab))

L55 ab
= L22 ((ab)(ab))((ab)(ab))
= L54 ((ab)(ba))((ab)(ab))
= L52 ((ba)(ba))((ab)(ab))

L56 a(b(bb))
= L53 a((bb)((ab)(ab)))
= L42 a(((ab)(ab))(bb))
= L40 a(((ab)(ab))(((bb)(ab))((bb)(ab))))
= L53 a((ab)((ab)(ab)))
= L42 a(((ab)(ab))(ab))
= L32 a(((ab)(ab))(a((ab)(ab))))
= L51 aa

T2 a(b(bb))
= L56 aa

L57 ((aa)(((ab)(ab))c))((aa)(((ab)(ab))c))
= L56 ((aa)(((ab)(d(dd)))c))((aa)(((ab)(d(dd)))c))
= L56 ((a(d(dd)))(((ab)(d(dd)))c))((a(d(dd)))(((ab)(d(dd)))c))
= L56 ((a(d(dd)))(((ab)(d(dd)))c))(d(dd))
= L42 (d(dd))((a(d(dd)))(((ab)(d(dd)))c))
= L42 (d(dd))((((ab)(d(dd)))c)(a(d(dd))))
= L42 ((((ab)(d(dd)))c)(a(d(dd))))(d(dd))
= L46 ((((ab)(d(dd)))c)(a(d(dd))))(((ab)(d(dd)))((d(dd))((ab)(d(dd)))))
= L33 ((((ab)(d(dd)))c)(a((a(d(dd)))a)))(((ab)(d(dd)))((d(dd))((ab)(d(dd)))))

= L33 ac

L62 (aa)b
= L61 b(((aa)b)((((aa)(((ac)(ac))d))b)(((aa)(((ac)(ac))d))b)))
= L59 b(((aa)b)((ab)(ab)))
= L50 b(((ab)(ab))((aa)(((aa)b)((ab)(ab)))))
= L45 b(((ab)(ab))((((ab)((bc)a))((ab)((bc)a)))(((aa)b)((ab)(ab)))))
= L59 b(ab)

L63 a(ab)
= L42 a(ba)
= L62 (bb)a

L64 a(bc)
= L45 ((a(bc))c)((ca)(a(bc)))
= L42 ((a(bc))c)((ca)((bc)a))
= L44 ((a(bc))c)a

L65 a(bc)
= L64 ((a(bc))c)a
= L42 a((a(bc))c)
= L42 a(c(a(bc)))

L66 ac
= L59 ((ac)(ac))((((ac)(ca))((ac)(ca)))b)
= L52 ((ac)(ac))((((ca)(ca))((ca)(ca)))b)
= L22 ((ac)(ac))((ca)b)

L67 (ab)(ab)
= L59 (((ab)(ab))((ab)(ab)))(((((ab)(ab))((ba)(ba)))(((ab)(ab))((ba)(ba))))c)
= L55 (((ab)(ab))((ab)(ab)))(((ba)(ba))c)
= L22 (ab)(((ba)(ba))c)

L68 a(((bc)(ba))((bc)(ba)))
= L42 (((bc)(ba))((bc)(ba)))a
= L63 a(a((bc)(ba)))
= L33 a(a((bc)(b((ba)b))))
= L33 a(a((bc)(((bc)(b((ba)b)))(bc))))
= A a((((bc)a)(b((ba)b)))((bc)(((bc)(b((ba)b)))(bc))))
= A a(b((ba)b))
= L33 a(ba)
= L62 (bb)a

L69 (bc)a
= L22 (((bc)(bc))((bc)(bc)))a
= L68 a(((((bc)(bc))((cb)(cb)))(((bc)(bc))a))((((bc)(bc))((cb)(cb)))(((bc)(bc))a)))
= L55 a(((cb)(((bc)(bc))a))((cb)(((bc)(bc))a)))

L70 (bc)a
= L69 a(((cb)(((bc)(bc))a))((cb)(((bc)(bc))a)))
= L67 a(((cb)(cb))((cb)(cb)))
= L22 a(cb)

L71 ((bc)(bc))a
= L68 a((((bc)(cb))((bc)a))(((bc)(cb))((bc)a)))
= L52 a((((cb)(cb))((bc)a))(((cb)(cb))((bc)a)))
= L66 a((cb)(cb))

L72 (ba)(((bc)a)((bc)a))
= L3 (a((ba)(((bc)a)((bc)a))))((aa)(((aa)((ba)(((bc)a)((bc)a))))(aa)))
= L33 (a((ba)(((bc)a)((bc)a))))((aa)((ba)(((bc)a)((bc)a))))
= L61 (ba)((aa)((ba)(((bc)a)((bc)a))))

(bc))
= L63 ((bc)(bc))(((bc)(bc))((((bc)a)(ab))(((bc)a)(ab))))
= L76 (((((bc)a)(ab))(((bc)a)(ab)))((bc)(ab)))((bc)(bc))
= L70 ((bc)(bc))(((bc)(ab))((((bc)a)(ab))(((bc)a)(ab))))
= L73 ((bc)(bc))(((bc)(bc))(ab))
= L63 ((ab)(ab))((bc)(bc))
= L47 ((ab)(ab))((c((ab)(bc)))(c((ab)(bc))))
= L75 ((ab)(ab))c
= L71 c((ba)(ba))

L79 a(c((ab)(ab)))
= L42 a(c((ba)(ba)))
= L78 a(((bc)(bc))a)
= L62 (((bc)(bc))((bc)(bc)))a
= L22 (bc)a

L80 a((ba)c)
= L70 (c(ba))a
= L79 a((ba)((ac)(ac)))
= L42 a(((ac)(ac))(ba))
= L77 a(((ac)(ac))(bb))
= L78 a(c((a(bb))(a(bb))))
= L79 ((bb)c)a

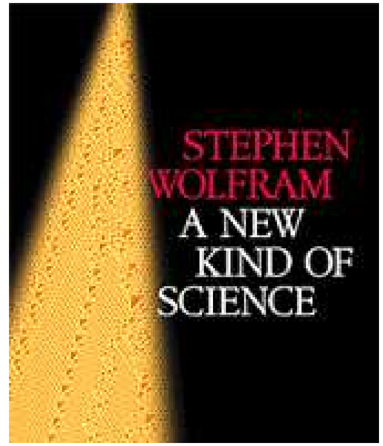
L81 ((ca)(ab))((ca)(ab))
= L40 ((((ca)(ab))((ca)(ab)))(((ca)(ab))((ca)(ab))))((aa)(((ca)(ab))((ca)(ab))))
= L75 ((((ca)(ab))((ca)(ab)))(((ca)(ab))((ca)(ab))))((aa)(ca))
= L22 ((ca)(ab))((aa)(ca))
= L70 ((aa)(ca))((ab)(ca))
= L40 a((ab)(ca))
= L70 ((ca)(ab))a
= L79 a((ab)((a(ca))(a(ca))))
= L70 a(((a(ca))(a(ca)))(ba))
= L77 a(((a(ca))(a(ca)))(bb))
= L78 a((ca)((a(bb))(a(bb))))
= L79 ((bb)(ca))a

T3 ((bb)a)((cc)a)
= L42 ((bb)a)(a(cc))
= L42 (a(cc))((bb)a)
= L22 (((aa)(aa))(cc))((bb)a)
= L80 ((bb)a)(((aa)((bb)a))(cc))
= L70 ((cc)((aa)((bb)a)))((bb)a)
= L81 (((aa)((bb)a))(((bb)a)c))(((aa)((bb)a))(((bb)a)c))
= L40 (a(((bb)a)c))(a(((bb)a)c))
= L80 ((((bb)(bb))c)a)((((bb)(bb))c)a)
= L42 ((c((bb)(bb)))a)((c((bb)(bb)))a)
= L40 ((((c((bb)(bb)))(c((bb)(bb))))((bb)(c((bb)(bb)))))a)((((c((bb)(bb)))(c((bb)(bb))))((bb)(c((bb)(bb)))))a)
= L60 ((((c((bb)(bb)))(c((bb)(bb))))b)a)((((c((bb)(bb)))(c((bb)(bb))))b)a)
= L42 ((b((c((bb)(bb)))(c((bb)(bb)))))a)((b((c((bb)(bb)))(c((bb)(bb)))))a)
= L78 ((((cb)(cb))((bb)(bb)))a)((((cb)(cb))((bb)(bb)))a)
= L40 ((((cb)(cb))((b((bb)(cb)))(b((bb)(cb)))))a)((((cb)(cb))((b((bb)(cb)))(b((bb)(cb)))))a)
= L65 ((((cb)(cb))((b((cb)(b((bb)(cb)))))(b((cb)(b((bb)(cb)))))))a)((((cb)(cb))((b((cb)(b((bb)(cb)))))(b((cb)(b((bb)(cb)))))))a)
= L75 ((((cb)(cb))b)a)((((cb)(cb))b)a)
= L78 ((b((cb)(cb)))a)((b((cb)(cb)))a)
= L31 ((cb)a)((cb)a)
= L70 (a(bc))(a(bc))

A proof that the axiom system $\{((b\circ c)\circ a)\circ(b\circ((b\circ a)\circ b))=a\}$ given as example (g) on page 808 can reproduce the Sheffer axiom system (c), and is thus a complete axiom system for logic. The proof involves taking the original axiom A and using it to establish a sequence of lemmas Ln, from which it is eventually possible to prove the three Sheffer axioms Tn. In each part of the proof each line can be obtained from the previous one just as on page 775 by applying the axiom or lemma indicated. Explicit $\bar{\wedge}$ operators have been omitted to allow expressions to be printed more compactly. The proof shown takes a total of 343 steps, and involves intermediate expressions with as many as 128 NANDs. It is quite possible that the proof could be considerably shortened. Note that any proof can always be recast without lemmas, but will usually then be much longer.

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*] ...

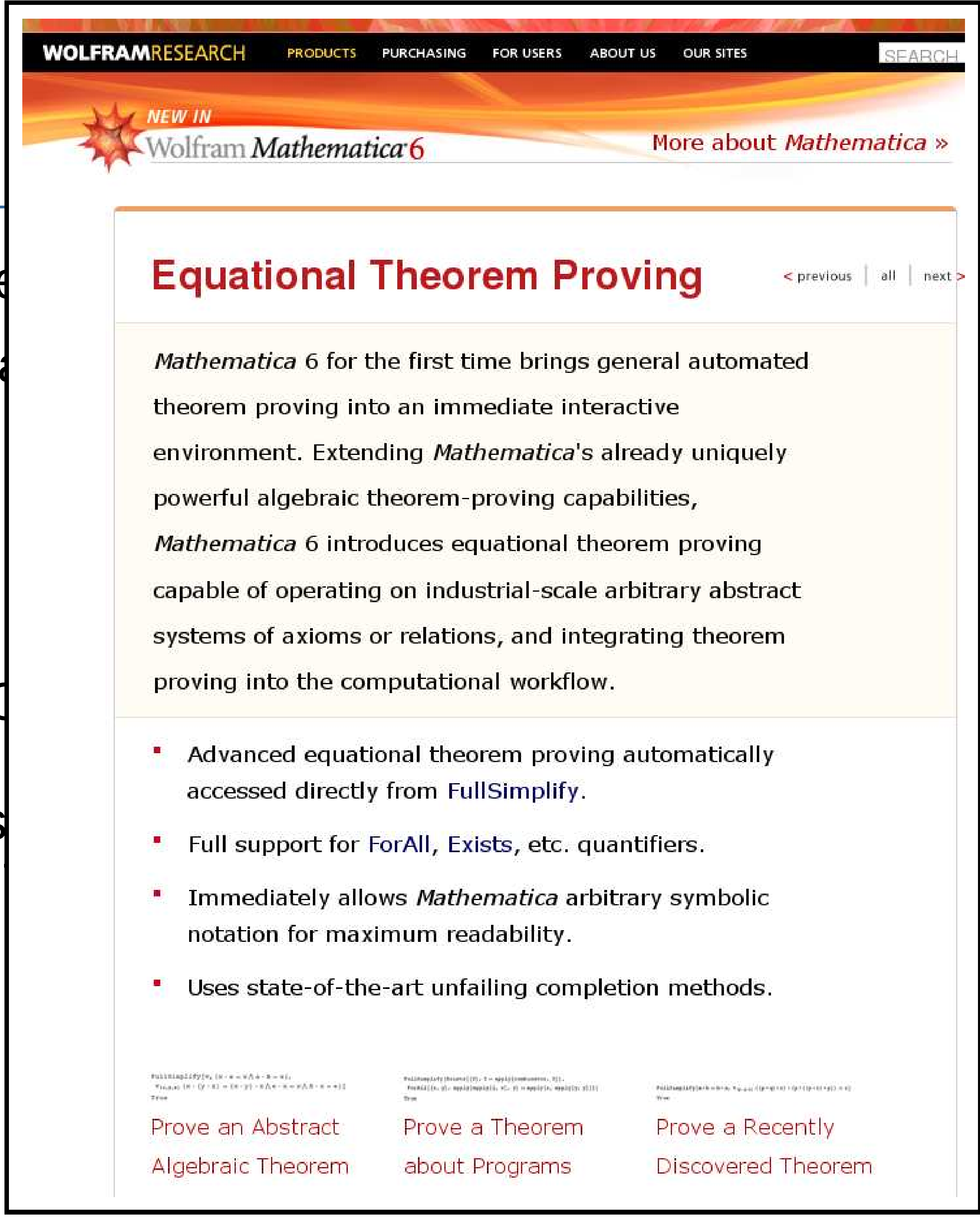
Integration i

- *Consequence* of these e

  "We are interested in a
  MATHEMATICA." (Oct.

- Introduced SW enginee
  System had to become
  Patent attorneys of MPC

- Functionality *available* s
  Encapsulated within Fu



WOLFRAMRESEARCH PRODUCTS PURCHASING FOR USERS ABOUT US OUR SITES SEARCH

NEW IN Wolfram *Mathematica* 6

More about *Mathematica* »

Equational Theorem Proving

< previous | all | next >

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.

Prove an Abstract Algebraic Theorem

Prove a Theorem about Programs

Prove a Recently Discovered Theorem

Integration into MATHEMATICA

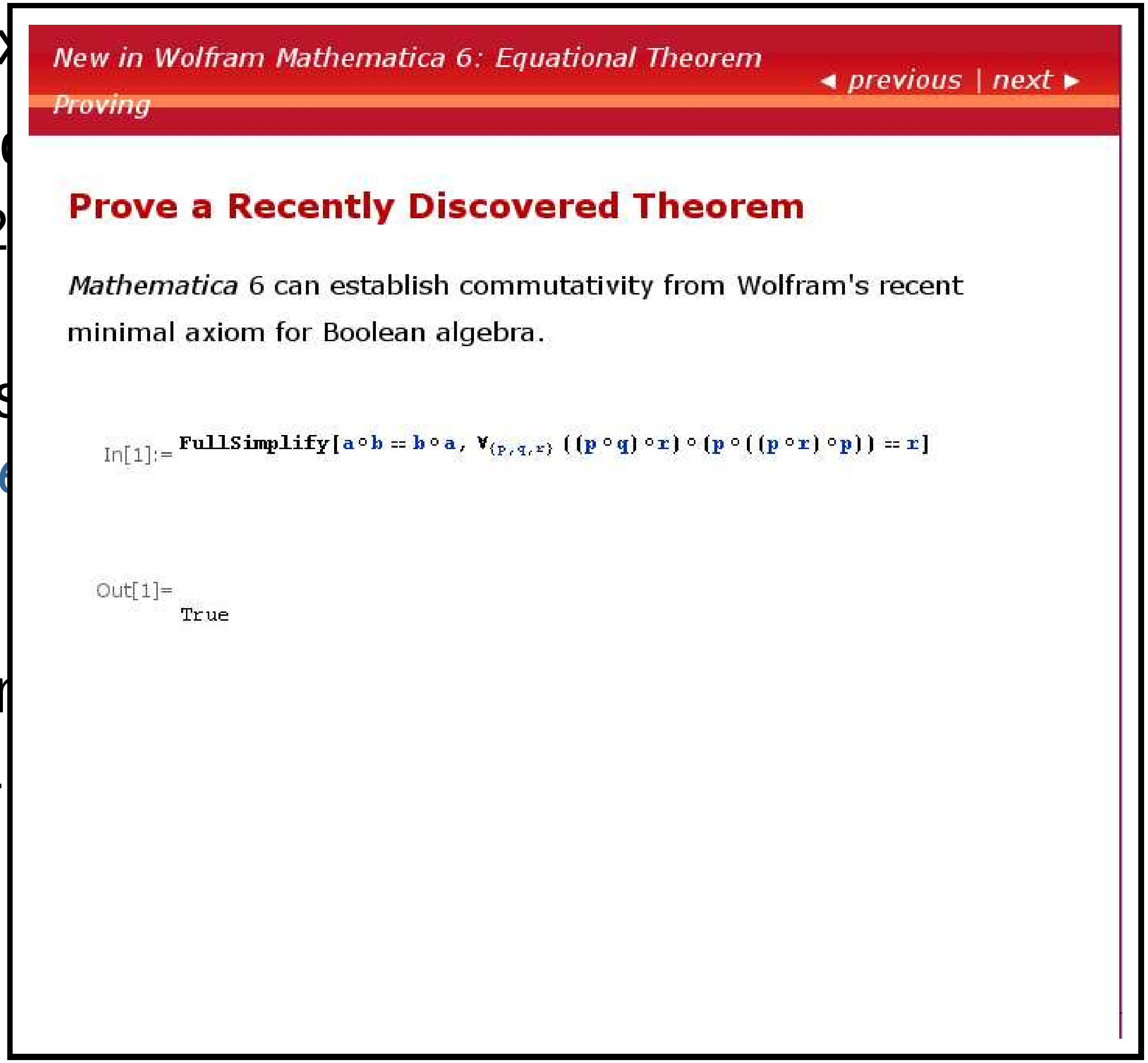
- *Consequence* of these ex

  "We are interested in a

  MATHEMATICA." (Oct. 2

- Introduced SW engineers
  System had to become *r*
  Patent attorneys of MPG

- Functionality *available* si
  Encapsulated within `Ful`



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)

Bachmair, Dershowitz, Hsiang 1986: Orderings for equational proofs. LICS-1.

Birkhoff 1935: On the structure of abstract algebras. Proc. Cambridge Phil. Soc. 31.

Christian 1989: Fast Knuth-Bendix completion: summary. RTA-3.

Comon, Narendran, Nieuwenhuis, Rusinowitch 1998: Decision problems in ordered rewriting. LICS-13.

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.

Hsiang, Rusinowitch 1987: On word problems in equational theories. ICALP-14.

References (2)

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.

KKP07: The structure of F-quasigroups. J Alg 317.

Knuth, Bendix 1970: Simple word problems in universal algebras. In Leech: Computational Problems in Abstract Algebra.

Lankford 1975: Canonical inference. ATP-32, UT Austin.

Lankford, Ballantyne 1977: Decision procedures for simple equational theories with commutative-associative axioms. ATP-39, UT Austin.

Martin, Nipkow 1990: Ordered rewriting and confluence. CADE-10.

McCune 2008: PROVER9 manual. `www.prover9.org`.

Newman 1942: On theories with a combinatorial definition of "equivalence". Annals of Mathematics 43(2).

References (3)

Nieuwenhuis, H., Riazanov, Voronkov 2001: On the evaluation of indexing techniques for theorem proving. IJCAR-1.

Nieuwenhuis, Rubio 2001 HAR: Paramodulation-based theorem proving. In Robinson, Voronkov: Handbook of Automated Reasoning.

Nieuwenhuis, Rivero 2002: Practical algorithms for deciding path ordering constraint satisfaction. I&C 178(2).

Overbeek 1971: A New Class of Automated Theorem-Proving Algorithms. PhD thesis, Penn State.

Peterson, Stickel 1981: Complete sets of reduction for some equational theories. JACM 28.

Phillips, Stanovský 2008: Automated theorem proving in loop theory. ESARM 2008.

SLOTHROP 2006 RTA: Wehrman, Stump, Westbrook: SLOTHROP: Knuth-Bendix completion with a modern termination checker. RTA-17.

References (4)

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

Stump, Tan 2005 RTA: The algebra of equality proofs. RTA-16.

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

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

Wolfram 2002: A New Kind of Science. Wolfram Media.

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).