Mechanical proofs of the Levi commutator problem

Maria Paola Bonacina
Dept. of Computer Science
The University of Iowa

July 1998

Outline

Problem formulation

The Argonne prover EQP

EQP experiments

Tuning parameters:

- to find a proof
- to improve performance

Proof presentation

The distributed prover Peers-mcd

Peers-mcd experiments

Discussion

Levi commutator problem

Group axioms:

$$e * x = x \quad \text{(left unit)}$$

$$x^{-1} * x = e \quad \text{(left inverse)}$$

$$(x * y) * z = x * (y * z) \quad \text{(associativity)}$$

Definition of commutator:

$$[x, y] = x^{-1} * y^{-1} * x * y$$

Theorem:

$$x * [y, z] = [y, z] * x$$

$$\Longleftrightarrow$$

$$[[x, y], z] = [x, [y, z]]$$

Formulation for the provers

Axioms:

$f(e, x) = x$ (left unit)

$f(g(x), x) = e$ (left inverse)

$f(f(x, y), z) = f(x, f(y, z))$ (associativity)

Definition of commutator:

$h(x, y) = f(g(x), f(g(y), f(x, y)))$

Theorem: the $\Rightarrow$ half

$f(x, h(y, z)) = f(h(y, z), x)$

$h(h(a, b), c) \;!=\; h(a, h(b, c))$

The $\Leftarrow$ half:

$h(h(x, y), z) = h(x, h(y, z))$

$f(a, h(b, c)) \;!=\; f(h(b, c), a)$

The $\Rightarrow$ half:

Otter 3.0.4

auto mode

0.07 sec

The <= half:

no fully automated Otter proof

Ordering-based strategies

Work on a set of clauses

Well-founded ordering on clauses
(complete simplification ordering)

Inference system:

expansion inference rules
(generate and add clauses)

contraction inference rules
(delete or reduce clauses)

Search plan:

no backtracking

indexing

mostly forward reasoning

Contraction-based strategies

Ordering-based strategies

with:

contraction inference rules

eager-contraction search plan.

Resolution
paramodulation
paradigm

Term rewriting
Knuth-Bendix
paradigm

⇘ ⇙

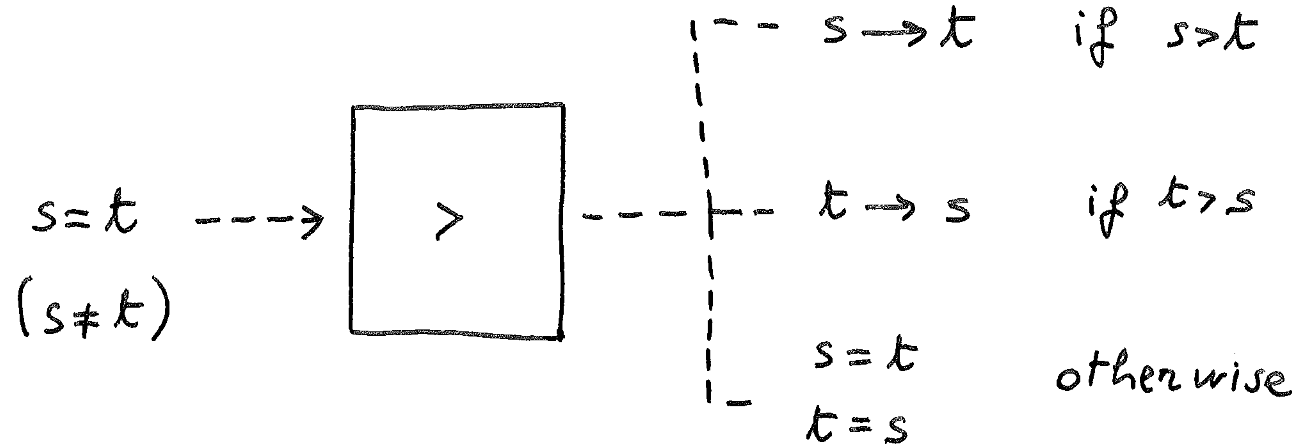
Ordering
based
strategies

EQP

Contraction-based strategies for equational reasoning with AC built-in.

Recursive path ordering:

- total precedence
- lexicographic / multiset status (default)



Inference rules

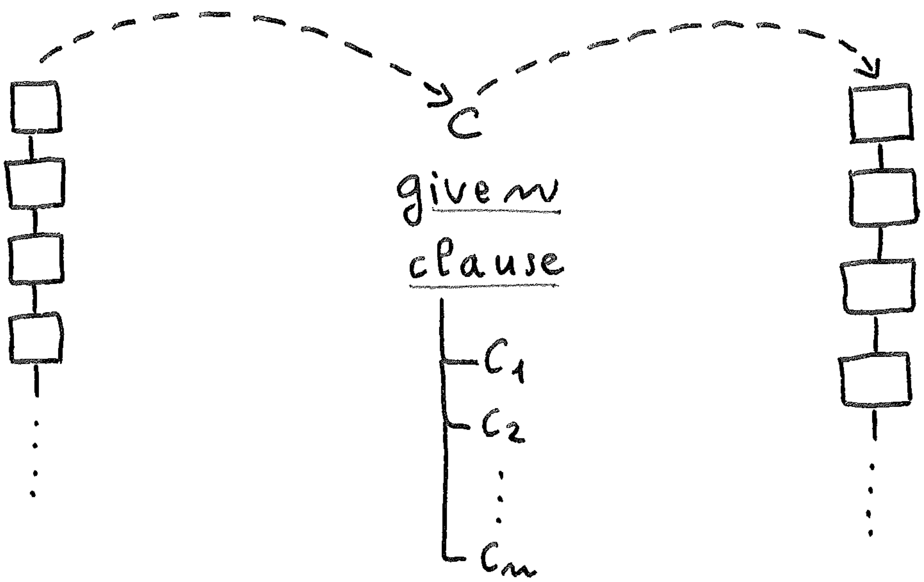
	ON/OFF (default)
Paramodulation ✓	ON
Ordered	OFF
Blocking	OFF
Basic	OFF
Simplification ✓ (by rewrite rules)	ON
Subsumption ✓	ON
Functional subsumption	OFF
Deletion by weight ✓ (parameter max-weight default weight of a term : number of symbols)	OFF

Search plan

Given clause algorithm (default)

Two lists of clauses:

Sos (Set of Support) (to be selected) — **Usable** (already selected)



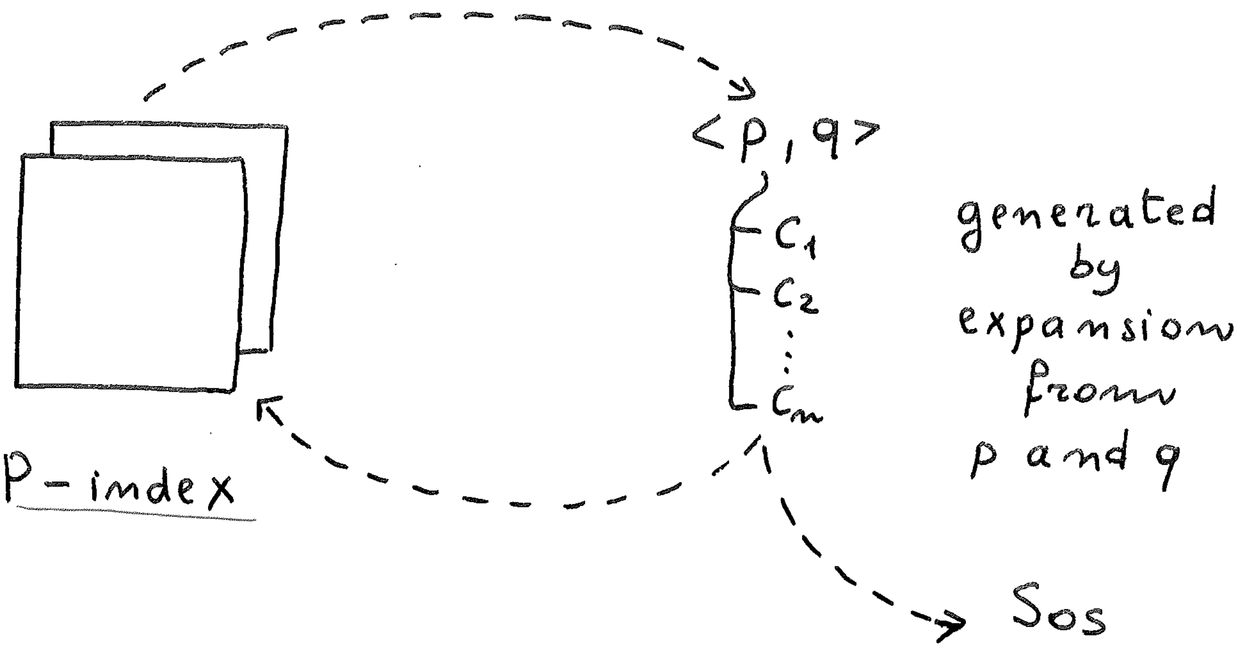
generated by expansion from c and clauses in Usable

Search plan

Pair algorithm

Sos and Usable

Index of pairs of clauses:



<p,q>: at least one in Sos

Search plan

Selection of given clause or next pair: smallest weight

Sos is kept sorted by weight

P-index returns a lightest pair not selected before

⇓

Best-first search

with weight as heuristic evaluation function

Search plan

pick-given-ratio = k

Select oldest (instead of lightest)
clause in Sos or
pair in P-index
every k selections

⇓

add some
breadth-first search

Search plan

Forward contraction:

normalize new clauses
with respect to the existing set.

Backward contraction:

keep set normalized
with respect to insertions.

Eager forward contraction

Eager backward contraction

	Otter	EQP
Logic	FOL + =	=
AC	NO	YES
Refinements of paramodulation	NO	YES
Simplification by equations	YES	NO
Given clause algorithm	YES	YES
Pair algorithm	NO	YES
Eager forward contraction	YES	YES
Eager backward contraction	NO	YES
Structure sharing	YES	NO
Flipping	NO	YES

Input for EQP

```
set(lrpo).
lex([a,b,c,e,f(x,x),g(x),h(x,x)]).
set(para_pairs).
assign(max_mem, 80000).
assign(max_weight, 49).
assign(pick_given_ratio, 4).
end_of_commands.

list(sos).
f(e,x) = x.
f(g(x),x) = e.
f(f(x,y),z) = f(x,f(y,z)).
h(x,y) = f(g(x),f(g(y),f(x,y))).
h(h(x,y),z) = h(x,h(y,z)).
f(a,h(b,c)) != f(h(b,c),a).
end_of_list.
```

Results

Time to □	148.96 sec ◄
Wall-clock time	163 sec ◄
Equations generated	96,219
Equations kept	9,657

With group axioms in Usable:

Time to □	127.77 sec ◄
Wall-clock time	145 sec ◄
Equations generated	96,846
Equations kept	9,854

(workstation HP B132L+ with 256M)

Parameter max-mem

How many Kbytes EQP is allowed to allocate dynamically

assign (max-mem, 80000).   high

With max-weight = 49 :

1st proof uses 22,949 Kbytes

2nd proof uses 23,437 Kbytes

With no max-weight :

out of memory even with 80,000

Parameter max-weight

O: out of memory

I: incomplete (halt without proof and we know it is a theorem)

P: proof

Rules of thumb: O → decrease

I → increase

max-weight	Axioms in Sos	Axioms in Usable
20	I	I
35	O	I
36	O	O
40	O	O
48	O	O
49	P	P

Improving performance

Raise max-weight : 60

	Axioms in Sos	Axioms in Usable
Time to □	60.28 sec	155.03 sec
Wall-clock time	64 sec	176 sec
Equations generated	32,553	75,534
Equations kept	4,491	9,490

Guided Otter proof

Use weight-list (purge-gen) with user-supplied patterns.

Precedence $a < b < c < e < h < f < g$

orients $f(g(x), f(g(y), f(x,y))) \rightarrow h(x,y)$

Time to □	316.08 sec
Wall-clock time	425 sec
Equations generated	871,524
Equations kept	6,806

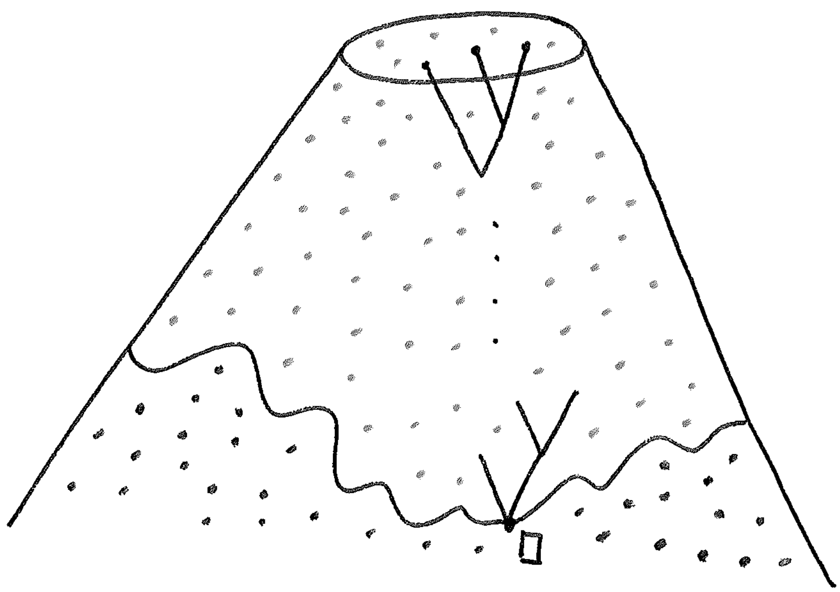
Mechanical proofs use pair algorithm search plan

Otter input

```
set(lrpo).
lex([a,b,c,e,h(x,x),f(x,x),g(x)]).
assign(max_mem, 20000).
assign(max_weight, 20).
assign(pick_given_ratio, 4).
list(usable).
x = x.
f(e,x) = x.
f(g(x),x) = e.
f(f(x,y),z) = f(x,f(y,z)).
end_of_list.
list(sos).
h(x,y) = f(g(x),f(g(y),f(x,y))).
h(h(x,y),z) = h(x,h(y,z)).
f(a,h(b,c)) != f(h(b,c),a).
end_of_list.
weight_list(purge_gen).
weight(h($(0),f($(0),h($(0),$(0)))), 100).
...
end_of_list.
```

Proof presentation

Proof : ancestor-graph of □



Proof reconstruction

Identifier and justification for each clause

```
1(wt=11)[flip(1)]
f(h(b,c),a)!=f(a,h(b,c)).
2(wt=5)[]f(e,x)=x.
3(wt=6)[]f(g(x),x)=e.
4(wt=11)[]f(f(x,y),z)=f(x,f(y,z)).
5(wt=13)[]
h(x,y)=f(g(x),f(g(y),f(x,y))).
6(wt=23)
[back_demod(1),demod([5,4,4,4,5])]
f(g(b),f(g(c),f(b,f(c,a))))!=
f(a,f(g(b),f(g(c),f(b,c)))).
...
9(wt=8)[para(3,4),demod([2]),flip(1)]
f(g(x),f(x,y))=y.
10(wt=6)[para(2,9)]f(g(e),x)=x.
...
```

Proof length

Axioms in Sos max-weight 49	123	(163 sec) (96,219)
Axioms in Usable max-weight 49	193	(145 sec) (96,846)
Axioms in Sos max-weight 60	215	(64 sec) (32,553)
Axioms in Usable max-weight 60	281	(176 sec) (75,534)

Clause - Diffusion

Parallel search by N processes

N separate derivations
(only one needs to succeed)

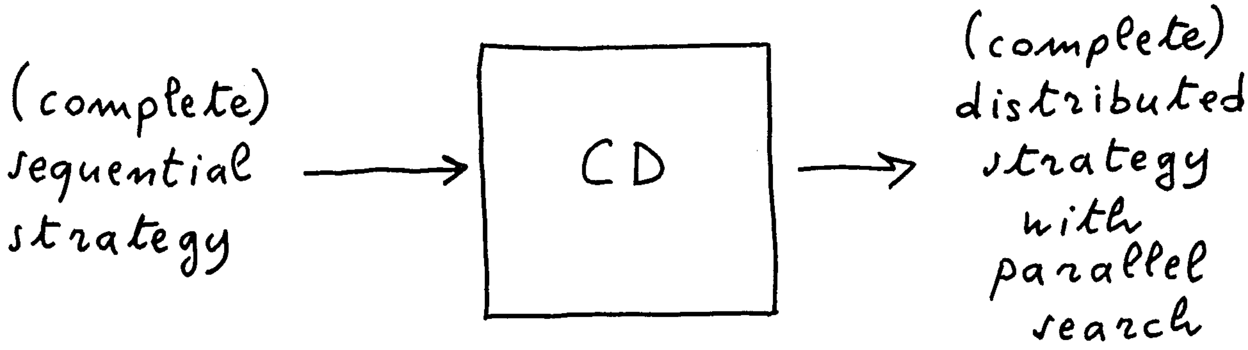
N separate data bases
(separate memories)

Subdivision of the search space

Communication

Possibly different search plans

The Clause-Diffusion methodology



Subdivision of the space:

Dynamic

Assign generated clauses to processes

Allocation algorithm

(logical, not physical allocation)

Subdivide inferences accordingly

e.g. paramodulation

backward simplification

The AGO criteria

Infinite search space of equations from input + inference systems

Search graph (hypergraph)

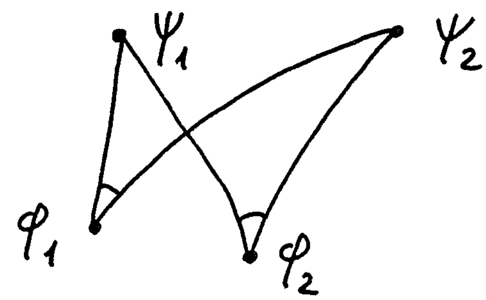
Finite ancestor-graphs

Use ancestor-graphs to assign equations to processes in such a way to limit overlap in an intuitive sense

The AGO criteria "parents"

Idea: proximity of equations in space

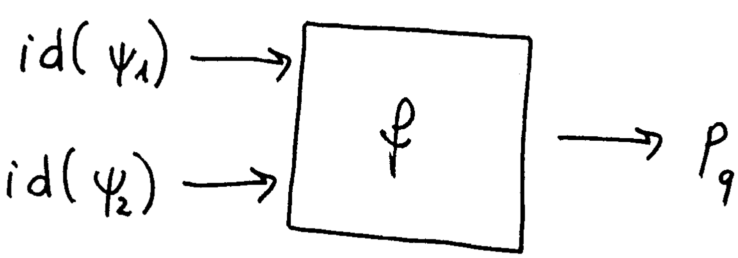
Example:



φ_1 to p_k

φ_2 to p_h

$\Rightarrow$ increase overlap of p_k and p_h



owner of φ_1, φ_2

- Various f
- Various notions of "parents"

The AGO criteria "parents"

Para-parents:

$id(\psi_1) + id(\psi_2) \mod N$

if paramodulation

0 otherwise

All-parents:

$id(\psi_1) + id(\psi_2) \mod N$

if paramodulation

$id(\psi) \mod N$

if backward-simplification

0 otherwise

Peers-mcd

EQP → [CD] → Peers-mcd

Equational logic with AC built-in

Contraction-based strategies

C and MPI

Networks of workstations

Multiprocessors

Results

Axioms in Usable
max-weight 49

	EQP	2-Peers
Time to □	127.77 sec	71.43 sec
Wall-clock time	145 sec	88 sec
Equations generated	96,846	38,126
Equations kept	9,854	7,348
Proof length	193	123

Speed-up = 1.65

Efficiency = 0.82

Results

Axioms in Sos

max-weight 60

	EQP	2-Peers
Time to □	60.28 sec	22.51 sec
Wall-clock time	64 sec	27 sec
Equations generated	32,553	18,374
Equations kept	4,491	2,831
Proof length	215	88

Speed-up = 2.37

Efficiency = 1.18

Discussion

Sequential experiments:

- search plan
  (given clause vs. pair)
- deletion by weight
- eager backward contraction

Distributed experiments:

- more processes did not improve
- search plan
  (other 3 subdivision criteria
  did not succeed)