Distributed theorem proving by Clause-Diffusion: the Peers-mcd prover

Maria Paola Bonacina

Dept. of Computer Science

The University of Iowa

Colloquium given at TU Dresden in May 1999

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

Examples

$\neg P \vee Q \qquad P \vee R$

$Q \vee R$

$$\frac{\{\neg P \vee Q,\ P \vee R\} \cup S}{\{\neg P \vee Q,\ P \vee R,\ Q \vee R\} \cup S}$$

$P(f(f(a))) \qquad f(x) = x$

$P(f(a))$

$P(a)$

$$\frac{\dfrac{\{P(f(f(a))),\ f(x) = x\} \cup S}{\{P(f(a)),\ f(x) = x\} \cup S}}{\{P(a),\ f(x) = x\} \cup S}$$

Theorem Proving

Complete inference system

Combinatorial explosion

1) Redundancy control
   (contraction, restriction to expansion)
2) Search plans
   (fair but not exhaustive)

Some of my directions of research

A) Distributed Deduction (1) (2)
   - Clause - Diffusion
   - Modified Cl. - Diffusion

B) Forward + Backward Reasoning (1) (2)
   - Target - oriented completion
   - Lemmatization

C) Strategy Analysis (1) (2)
   - Contraction - based
   - Distributed - search contraction - based

Motivation

For Distributed Deduction:

Improve performance by distributed search

Improve applicability to large problems

Study new forms of reasoning and search

For Contraction-based Strategies:

Equational reasoning

Behave well sequentially:
OTTER, RRL, REVEAL, SPASS, EQP, GANDALF ...

Parallelization is challenging

Background: parallelism and deduction

Fine-grain parallelism

one search process

sequential inferences

parallel algorithms

e.g. parallel rewriting

Medium-grain parallelism

one search process

parallel inferences

Coarse-grain parallelism

parallel searches

The problem: Subdivision of the search space

Parallel search
- Subdivide space (Distributed search)
- use different search plans (Multi-search)
- ....

Subdivide space:

- one search space
  (input spec. + inference rules)
- many search processes
- limit their overlap
- infinite space / partial dynamic knowledge

Outline

Clause - Diffusion

Modified Clause - Diffusion

Ancestor - Graph Oriented (AGO)
heuristic criteria for subdivision
of search space

The distributed theorem prover Peers - mcd
(Modified Clause - Diffusion + EQP)

Experiments

Robbins Algebra

Levi Commutator Problem

Analysis of experiments

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

(complete) sequential strategy $\longrightarrow$ $\boxed{\text{CD}}$ $\longrightarrow$ (complete) distributed strategy with parallel search

Subdivision of the space:

- Dynamic
- Assign generated clauses to processes

  Allocation algorithm

  (logical, not physical allocation)

- Subdivide inferences accordingly

  e.g. paramodulation

  backward simplification

Evolution of Clause-Diffusion and Clause-Diffusion provers

Clause-Diffusion 1992
(STACS 1993, FI 1995)

Modified Clause-Diffusion 1994
(PASCO 1994, JSC 1996)

Aquarius (Otter 2.2) 1992
(C + PCN)
(DISCO 1993, JSC 1995)

CD

Peers (OPS) 1993 - 1994
(C + P4)
(CADE 1994)

Peers-mcd (OPS) 1995
(C + P4)

MCD

Peers-mcd (EQP 0.9) 1996-1998
(C + MPI)
(CADE 1997, PASCO 1997, CADE 1998)

Peers-mcd (EQP 0.9c, 0.9d) 1999
(C + MPI)

MCD successor of CD

Better trade-off between parallelism and redundancy control:
subdivides also backward contraction,
less duplication without losing contraction power,
uniform treatment of raw clauses.

Better communication scheme:
fewer message types,
fewer messages.

Distributed proof reconstruction:
unambiguous naming scheme,
comprehensive and safe communication scheme.

The AGO criteria

Infinite search space of equations from input + inference systems

Search graph (hypergraph)

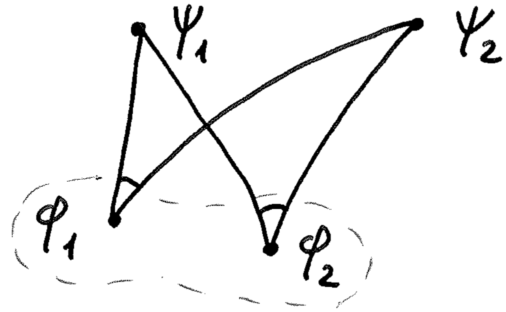
Finite ancestor-graphs

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:



$$\left.\begin{array}{l}\varphi_1 \text{ to } p_k \\ \varphi_2 \text{ to } p_h\end{array}\right\} \Rightarrow \text{increase overlap of } p_k \text{ and } p_h$$

$$\begin{array}{l} id(\psi_1) \longrightarrow \\ id(\psi_2) \longrightarrow \end{array} \boxed{\;f\;} \longrightarrow p_q \quad \text{owner of } \varphi_1, \varphi_2$$

- 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

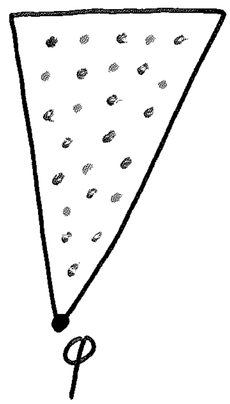
0 otherwise

The AGO criterion "majority"

Assign φ to P_k active near φ
(proximity of equations and processes)

P_k

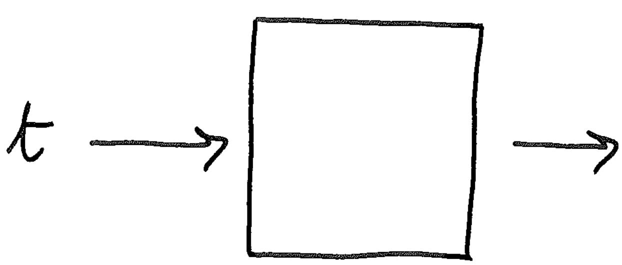
P_h



ancestor-graph of φ

φ to $P_k \Rightarrow$ increase overlap

Thus: φ to P_h



P_q owns majority in t

Strategies in Peers-mcd

Inference system – n – search plan

Default inference system:

(AC)- paramodulation
(AC)- simplification
subsumption

Practical feature: deletion by weight

Search plan:

- "given clause" algorithm
- "pairs" algorithm

Parameter n: if set, do breadth-first selection instead of best-first selection every n selections.

Default: always best-first

Strategies in the experiments

For each problem pick strategy that did best sequentially.

"Start-n-pair":

AC-paramodulation AC-simplification subsumption deletion by weight	inference system
"pairs" algorithm best-first search	search plan

Best "complete" sequential strategy

Basic-n-pair Superϕ-n-pair	better on some problems "more incomplete"

Subdivision criteria

AGO:

para-parents

all-parents

majority

Un-informed:

rotate

Others:

syntax

Experiments

For most experiments there exists an AGO criterion which leads Peers-mcd to speed-up over EQP

For most experiments with strategy start-n-pair there is an AGO criterion which enables some configuration of Peers-mcd to obtain super-linear speed-up

Fastest known proofs of three hard lemmas in Robbins algebra.

First mechanical proof (fully automated) of the Levi Commutator problem.

Experiments

A classical problem in ring theory

$x^3 = x$ implies commutativity

Strategy	*Criterion*	*EQP0.9*	*1-Peers*	*2-Peers*	*4-Peers*
super0-n-pair	rotate	41	42	16	10
super0-n-pair	para-parents	41	42	15	**5**
super0-n-pair	all-parents	41	42	14	**5**
super0-n-pair	majority	41	42	19	7

4-Peers: speed-up = 8.2 efficiency = 2

Strategy	*Criterion*	*EQP0.9*	*1-Peers*	*2-Peers*	*4-Peers*
start-n-pair	rotate	143	133	137	50
start-n-pair	para-parents	143	133	86	**22**
start-n-pair	all-parents	143	133	84	65
start-n-pair	majority	143	133	113	48

4-Peers: speed-up = 6.5 efficiency = 1.6

Max-weight = 60 for all processes

(HP 715 , 64M)

Robbins algebra

$$\left.\begin{array}{l}\text{Huntington axiom}\\ \text{AC of } +\end{array}\right\}\ \text{Boolean algebra}$$

$$\left.\begin{array}{l}\text{Robbins axiom}\\ \text{AC of } +\end{array}\right\}\ \text{Robbins algebra}$$

$$\text{Robbins axiom} \xrightarrow[1933]{?} \text{Huntington axiom}$$

Yes: EQP 1996

$$\begin{array}{c}\text{Robbins axiom}\\ \downarrow\\ \text{Second Winker Condition}\\ \downarrow\\ \text{First Winker Condition}\\ \downarrow\\ \text{Huntington axiom}\end{array}$$

Robbins algebra

Huntington axiom:

$$n(n(x)+y) + n(n(x)+n(y)) = x$$

Robbins axiom:

$$n(n(x+y)+n(x+n(y))) = x$$

First Winker Condition (FWC):

$$\exists x \exists y \quad x+y = x$$

Second Winker Condition (SWC):

$$\exists x \exists y \quad n(x+y) = n(x)$$

Another formulation: FWC implies $\exists x \; x+x=x$

Strategy	*Criterion*	*EQP0.9*	*1-Peers*	*2-Peers*	*4-Peers*	*6-Peers*
start-n-pair	rotate	3,705	3,953	1,349	1,340	1,631
start-n-pair	para-parents	3,705	3,953	933	915	**522**
start-n-pair	all-parents	3,705	3,953	1,227	851	608
start-n-pair	majority	3,705	3,953	997	1,043	1,187

6-Peers : speed-up = 7 efficiency = 1.2

Strategy	*Criterion*	*EQP0.9*	*1-Peers*	*2-Peers*	*4-Peers*	*6-Peers*
basic-n-pair	rotate	1,661	1,617	1,566	1,646	1,563
basic-n-pair	para-parents	1,661	1,617	2,021	698	**551**
basic-n-pair	all-parents	1,661	1,617	1,598	700	**551**
basic-n-pair	majority	1,661	1,617	1,548	3,233	1,820

6-Peers: speed-up = 3 efficiency = 0.5

Max-weight = 30 for all processes

Lemma: FWC implies H

Strategy	*Criterion*	*EQP0.9*	*1-Peers*	*2-Peers*	*4-Peers*	*6-Peers*	*8-Peers*
start-n-pair	rotate	4,857	4,904	3,557	1,177	3,766	2,675
start-n-pair	para-parents	4,857	4,904	1,437	2,580	3,934	2,158
start-n-pair	all-parents	4,857	4,904	1,534	2,588	1,819	**519**
start-n-pair	majority	4,857	4,904	872	**709**	707	1,809

4-Peers: speed-up = 6.8 efficiency = 1.7

Another formulation: FWC implies $\exists y \forall x \; x+y=x$

Strategy	*Criterion*	*EQP0.9*	*1-Peers*	*2-Peers*
start-n-pair	rotate	3,649	3,809	2,220
start-n-pair	para-parents	3,649	3,809	1,591
start-n-pair	all-parents	3,649	3,809	1,086
start-n-pair	majority	3,649	3,809	**485**

2-Peers: speed-up = 7.5 efficiency = 3.7

Max-weight = 30 for all processes

Lemma : SWC implies FWC

Sequential time: almost 6 days

Max-weight = 34 for all processes

Strategy	*Criterion*	*EQP0.9*	*1-Peers*	*2-Peers*	*4-Peers*	*6-Peers*	*8-Peers*
start-n-pair	rotate	518,393	520,336	265,145	71,416	6,391	5,436
start-n-pair	para-parents	518,393	520,336	**10,162**	108,975	7,792	**3,283**
start-n-pair	all-parents	518,393	520,336	**10,023**	122,719	7,598	**3,357**
start-n-pair	majority	518,393	520,336	161,779	54,660	68,919	7,415

Most efficient: 2-Peers with all-parents

time: 2 hr 47' 3"

Speed-up: ~ 52

efficiency: ~ 26

Fastest proof: 8-Peers with para-parents

time: 0 hr 54' 43"

Speed-up: ~ 158

efficiency: ~ 20

Robbins axiom implies SWC

Another strategy: basic ϕ-4-pair

Sequential time: 1 day 17 hr 30' 4"

on 41 hr 30' 4"

2-Peers with majority:

23 hr 33' 26"

speed-up = 1.76

efficiency = 0.88

Max-weight = 50 for all processes

(SGI ONYX shared memory machine)

Levi commutator problem

Group axioms:

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

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

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

Definition of commutator:

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

Theorem:

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

$$\Longleftrightarrow$$

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

$\Rightarrow$: very easy (e.g., for Otter)

$\Leftarrow$: hard (no Otter fully automated proof)

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

(Workstation HP B132L+ with 256M)

Statistics

Clauses generated

Clauses kept

Demodulation time

normalization

forward simplification

backward simplification

Back-demod-find-time

time to find clauses that can be backward-simplified

Proofs

sequential / distributed

length

clauses in common

Example of statistics from a sequential and a distributed derivation

Lemma: FWC implies H

Peers-mcd: 4-Peers with majority

Statistics	*EQP0.9*	*Peer0*	*Peer1*	*Peer2*	*Peer3*	*Peers-mcd*
clauses generated	25,939	5,047	5,138	2,826	2,687	15,698
clauses kept	2,905	928	556	189	144	1,817
retention	11%	18%	11%	7%	5%	12%
proof found	1	0	0	1	0	1
proof length	107	N/A	N/A	123	N/A	123

Different proofs: 55 clauses in common

Times	*EQP0.9*	*Peer0*	*Peer1*	*Peer2*	*Peer3*
wall-clock-time	4,902	705	704	704	704
cpu-time	4,665.60	664.79	677.03	603.66	612.11
demodulation-time	3,557.55	375.26	381.78	294.59	314.55
back-demod-find-time	876.95	253.81	250.83	252.63	252.82

Max-weight = 30 for all processes

Example of statistics from a sequential and a distributed derivation

Lemma: SWC implies FWC

Peers-mcd: 2-Peers with all-parents

Statistics	*EQP0.9*	*Peer0*	*Peer1*	*Peers-mcd*
clauses generated	354,477	14,712	12,445	27,157
clauses kept	3,730	1,941	980	2,921
retention	1%	13%	8%	11%
proof found	1	1	0	1
proof length	24	24	N/A	24

Different proofs: 20 clauses in common

Times	*EQP0.9*	*Peer0*	*Peer1*
wall-clock-time	518,393	9,865	9,867
cpu-time	514,257.39	9,784.46	9,793.00
demodulation-time	510,740.25	8,072.46	8,169.17
back-demod-find-time	2,495.22	1,511.59	1,498.45

Max-weight = 34 for all processes

Analysis of experiments

Super-linear speed-up:
- much fewer clauses generated
- effective subdivision of the space

In some cases, e.g. SWC $\rightarrow$ FWC:
- higher % clauses kept
- same contraction
- search may be better focused

Contraction time:
- most of time for both EQP and Peers-mcd

Proofs: majority of equations in common
difference: parallel search

Scalability: size of problem
dynamic subdivision

Discussion and future work

Parallelism in theorem proving as a new form of search

High-performance theorem proving needs many tools: parallel search by distributed processes is one

Design / implementation:

- more AGO criteria
- more experiments
- proof checker
- tools for proof analysis / comparison

Theory:

- modelling parallel search
- strategy analysis