The Clause-Diffusion theorem prover Peers-mcd

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

July 1997

Peers-mcd: an overview

Distributed theorem prover

EQP + Clause Diffusion

Contraction-based strategies

Equational logic with AC

C and MPI

Network of workstations or multiprocessor

A tool for experimenting with Parallel Search

Fine-grain parallelism:

e.g. parallel rewriting

Medium-grain parallelism:

parallel inferences

Coarse-grain parallelism:

parallel search

Clause - Diffusion

theorem proving strategy → CD → distributed strategy

sequential search (complete)	parallel search (complete)

EQP → CD → Peers-mcd

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

Subdivision of the search space in Clause - Diffusion

Build dynamically a partition of the search space by

assign generated clauses to processes
allocation criterion
(logical not physical allocation)

subdivide inferences accordingly
(both expansion and contraction)

Subdivision of the search space in Peers-mcd

Intuition: limit the overlap of the parallel searches

Approach:

model the search space as a search graph (including contraction)

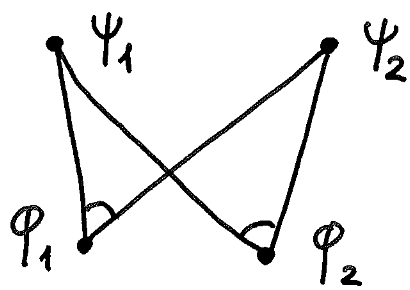
each clause has ancestor-graph

Ancestor-Graph Oriented (AGO) allocation criteria

AGO criteria

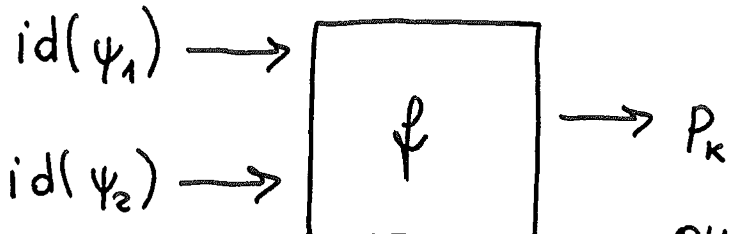
Intuition: notion of proximity of clauses in the search graph

AGO criterion "parents":



φ_1 and φ_2 are close

$$\left.\begin{array}{l} \varphi_1 \text{ to } P_i \\ \varphi_2 \text{ to } P_j \end{array}\right\} \Longrightarrow P_i \text{ and } P_j \text{ overlap}$$

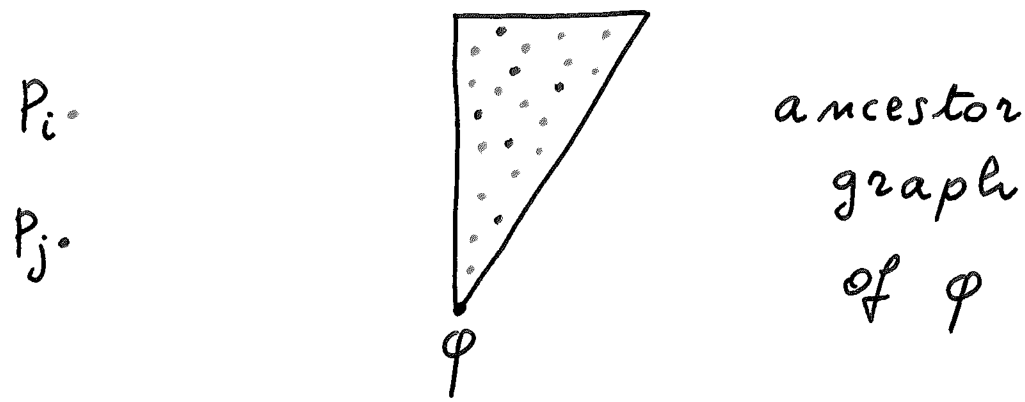


owner of φ_1 / φ_2

AGO criteria

Intuition: notion of proximity between clauses and processes

AGO criterion "majority":



φ is closer to p_i than p_j

φ assigned to p_j $\Rightarrow$ increase overlap of p_i and p_j

Assign φ to process that owns majority of ancestors

Robbins algebra

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

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

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

Yes: EQP 1996

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

A case study in Robbins algebra

First Winker Condition: $\exists x \exists y \; x + y = x$

Lemma: FWC implies Huntington

Strategy: start-n-pair

$$\left.\begin{array}{l} \text{AC- paramodulation} \\ \text{AC- simplification} \\ \text{subsumption} \\ \text{deletion by weight} \end{array}\right\} \text{inference system}$$

$$\left.\begin{array}{l} \text{pairs algorithm} \\ \text{best-first search} \end{array}\right\} \text{search plan}$$

Best "complete" sequential strategy on the problem

First formulation

$$\exists x \exists y \; x+y=x \longrightarrow \exists x \; x+x=x$$

$$(\exists x \; x+x=x \longrightarrow H \quad \text{in a few seconds})$$

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	parents	3,705	3,953	933	915	**522**
start-n-pair	majority	3,705	3,953	997	1,043	1,187

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

Max-weight = 30 for all processes

Network of workstations HP 715

Second formulation

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

$$(\exists y \forall x \quad x+y=x \quad \rightarrow \quad \exists x \quad x+x=x \quad \text{trivially})$$

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

2-Peers: speed-up = 7.5

efficiency = 3.7

Third formulation

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

Strategy	*Criterion*	*EQP0.9*	*1-Peers*	*2-Peers*	*4-Peers*
start-n-pair	rotate	4,857	4,904	3,557	1,177
start-n-pair	parents	4,857	4,904	1,437	2,580
start-n-pair	majority	4,857	4,904	872	**709**

4-Peers: speed-up = 6.8

efficiency = 1.7

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	parents	518,393	520,336	**10,162**	108,975	7,792	**3,283**
start-n-pair	majority	518,393	520,336	161,779	54,660	68,919	7,415

Most efficient: 2-Peers

time: 2 hr. 49' 22"

speed-up: 51

efficiency: 25.5

Fastest proof: 8-Peers

time: 0 hr 54' 43"

speed-up: ~ 158

efficiency: ~ 20

Final lemma:

Robbins axiom implies SWC

Another strategy: basic Ø-4-pair

Sequential time: 4 days 3 hr 24' 7"

Moving to a shared-memory machine:

sequential time: 1 day 17 hr 30' 4"

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

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

Analysis of experiments

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

In some cases, e.g. SWC → 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:

Practical theorem proving needs many tools: parallel search is one

Super-linear speed-up

Scalability

Future work:

Combine subdivision of space with use of different search plans

Strategy analysis