Distributed reasoning by Clause-Diffusion: the Peers-mcd.d prover

MARIA PAOLA BONACINA
DEPT. OF COMPUTER SCIENCE
THE UNIVERSITY OF IOWA

Outline

Motivation

Modified Clause-Diffusion

The Peers-mcd.d prover

Experiments

Discussion

Automated Reasoning

Study mechanical forms of logical reasoning

problem

$$\downarrow$$

$$H \overset{?}{\models} \varphi$$

ATP

$$H \cup \{\neg\varphi\} \vdash \bot$$

AMB

$$M \models H \cup \{\neg\varphi\}$$

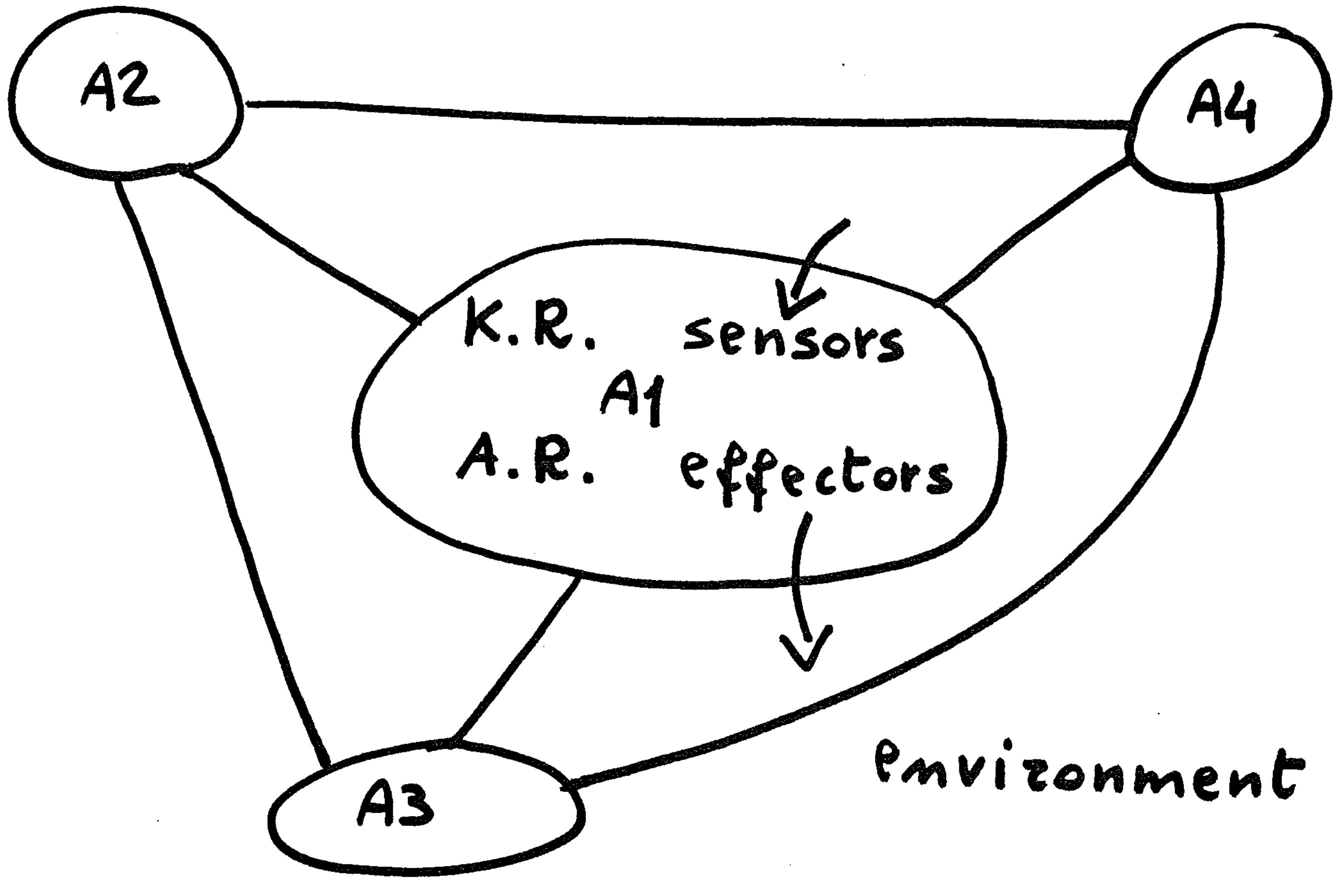
- HW/SW verification
- program generation
- intelligent agents

A.R. works: [1990-2000]

- Moufang identities in rings
- Axioms Lukasiewicz many-valued logic
- Single axioms for groups
- Robbins algebras are Boolean
- Program synthesis in astronomy
- Verification cryptographic protocols
- Verification message-passing Unix

Meanwhile:

multi-agent paradigm



"Disembodied" agent: A.R. program

Distributed reasoning

- More power:

  faster proofs (performance)

  more proofs (applicability)

- Study new forms of reasoning and search:

  search plan design

  multi-agent context

Research program

Center: Automated Reasoning

Emphasis: Control of deduction

Some directions:

* Combination of forward and backward reasoning, e.g.,

  Target-oriented equational reasoning

  Lemmatization in semantic strategies

* Distributed automated deduction, e.g.,

  Clause-Diffusion methodology

  Modified Clause-Diffusion

  AGO - criteria

  Combination of distributed and multi-search

  Systems: Aquarius, Peers, Peers-mcd. *

* Strategy analysis, e.g.,

  Search space reduction by contraction

  Distributed search for contraction-based strategies

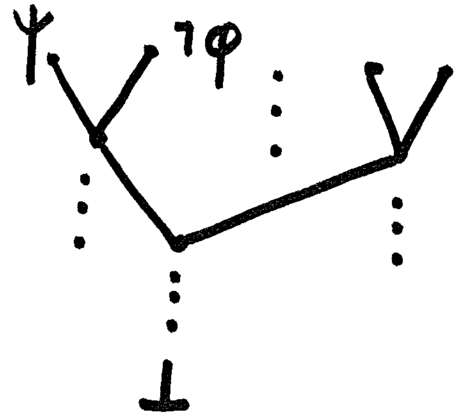
The Modified Clause-Diffusion methodology

ATP : inference + search problem

Input data: $S = H \cup \{\neg\varphi\}$

formulae, e.g., clauses

Desired output: a proof



refutation

Operations to get there:

inference rules

premises

$$\frac{\psi_1 \;\cdots\; \psi_n}{\psi}$$

consequence

Examples

I) $$\frac{\neg P \vee Q \qquad P \vee R}{Q \vee R}$$

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

II)
$$\begin{tikzcd}
P(f(f(a))) \arrow[d, ">"'] & f(x) = x \arrow[dl, dash] \arrow[ddl, dash] \\
P(f(a)) \arrow[d, ">"'] & \\
P(a) &
\end{tikzcd}$$

$$\begin{array}{c}
\{P(f(f(a))),\ f(x) = x\} \cup S \\
\hline
\{P(f(a)),\ f(x) = x\} \cup S \\
\hline
\{P(a),\ f(x) = x\} \cup S
\end{array}$$

I) Expansion rules

II) Contraction rules
($>$: well-founded ordering)

Theorem-proving strategy

$$\mathcal{C} = \langle I; \Sigma \rangle$$

I: inference system

Σ: search plan

- selection of premises
- selection of rules (e.g., eager contraction)

Derivation:

$$S_0 \vdash S_1 \vdash \ldots \vdash S_i \vdash S_{i+1} \vdash \ldots$$

Background: parallelism & deduction

Fine-grain parallelism
one search, sequential inferences
parallel inner algorithms

Medium-grain parallelism
one search, parallel inferences

Coarse-grain parallelism
many searches, parallel derivations

distributed search

multi-search

with homogeneous / heterogeneous I's

Modified Clause-Diffusion

$$\mathcal{C} = \langle I, \Sigma \rangle \longrightarrow \boxed{\text{MCD}} \longrightarrow \mathcal{C}' = \langle I, \Sigma' \rangle$$

complete sequential strategy		complete distributed strategy

Parallel search by N concurrent asynchronous, communicating processes.

Peer processes: no master-slaves.

N separate derivations:
only one needs to succeed.

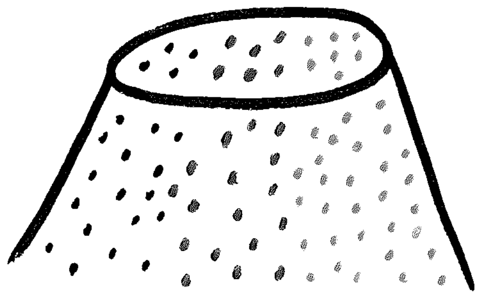
N separate databases:
separate memories $\Rightarrow$ no conflicts.

$$P_i : \; S^i_0 \vdash S^i_1 \ldots S^i_n \vdash S^i_{n+1} \ldots$$

$$i \in [0, N-1]$$

Distributed search in MCD

Subdivide search space:



MCD: dynamic partition of generated clauses $\Rightarrow$ subdivision of inferences

Every ψ is assigned to a unique p_i:

$\neg A \vee p = q$ "belongs" to p_i

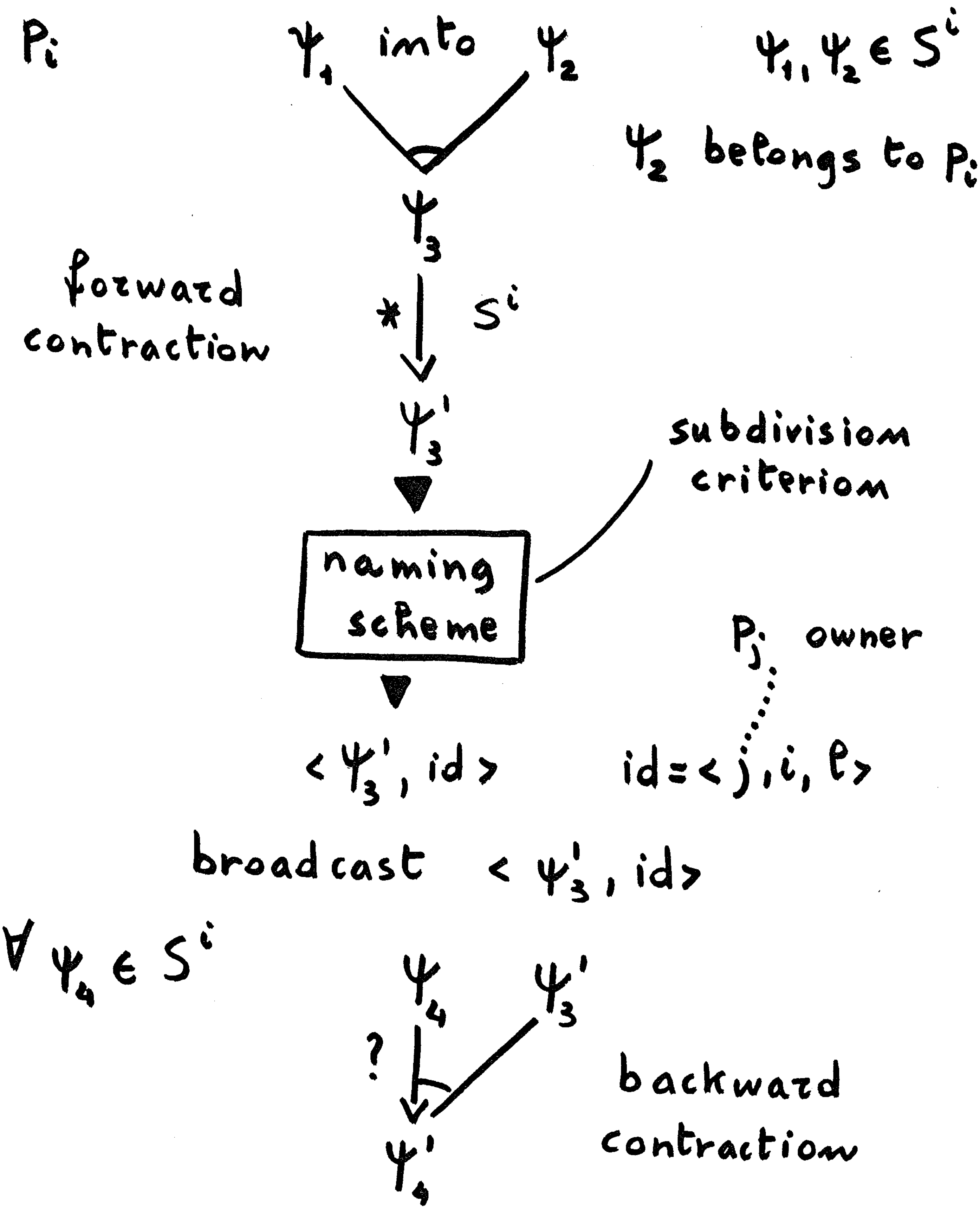
$$\neg A \vee p[s] = q \qquad s = t$$

$$t \not\geq s$$

$$\neg A \vee p[t] = q$$

allowed only to p_i

Generation / Diffusion of a clause



Remarks

- No need of master / scheduler for subdivision:
  every process subdivides the clauses it generates.

- No need of master for communication:
  asynchronous broadcasting.

- Forward / backward contraction:
  Keep each S^i inter-reduced.

What about backward-contraction?

Backward contraction

$$
\begin{array}{ccccc}
\langle \varphi, id \rangle & \ldots & \langle \varphi, id \rangle & \ldots & \langle \varphi, id \rangle \\
 & & \downarrow & & \\
 & & \langle \varphi', id' \rangle & & \\
P_0 & \ldots & P_j : \text{owner} & \ldots & P_{N-1}
\end{array}
$$

Too little contraction: redundancy

$$
\begin{array}{ccccc}
\langle \varphi, id \rangle & \ldots & \langle \varphi, id \rangle & \ldots & \langle \varphi, id \rangle \\
\downarrow & & \downarrow & & \downarrow \\
\langle \varphi', id'_0 \rangle & \ldots & \langle \varphi', id'_j \rangle & \ldots & \langle \varphi', id'_{N-1} \rangle \\
P_0 & & P_j & & P_{N-1}
\end{array}
$$

Redundancy by duplication +

ambiguous naming scheme

Solution in MCD

$\langle \varphi, id \rangle$ ... $\langle \varphi, id \rangle$ ... $\langle \varphi, id \rangle$

deleted

$$\langle \varphi, id \rangle \longrightarrow \langle \varphi', id' \rangle$$

deleted

P_0 ... P_j : owner ... P_{N-1}

Advantages:

- no redundancy by lack of contraction
- minimize redundancy by duplication
- unambiguous naming scheme
- uniform treatment of raw clauses

Fairness of distributed derivations

Refutational completeness of I +
Fairness of Σ' = Completeness of $\mathcal{C}'$

$\forall \bar{x}$ persistent non-redundant
$\forall f$ expansion rule
$\exists P_k$ such that

1) P_k has $\bar{x}$ (fairness of communication)
2) P_k is allowed to apply f to $\bar{x}$
(fairness of subdivision)
3) and all local derivations are fair

$\Rightarrow$ the distributed derivation is fair.

Th.: MCD satisfies (1), (2), (3).

Th.: if P_i generates $\square$ it can reconstruct the proof based on its final state

The Peers-mcd.d

prover

Major features

- Inference system:
  (AC) - paramodulation
  (AC) - simplification
  functional subsumption

  practical feature: deletion by weight

- AGO subdivision criteria

- Combination of distributed search and multi-search

The AGO criteria

Infinite search space of equations
from input + inference systems

Search graph (hypergraph)

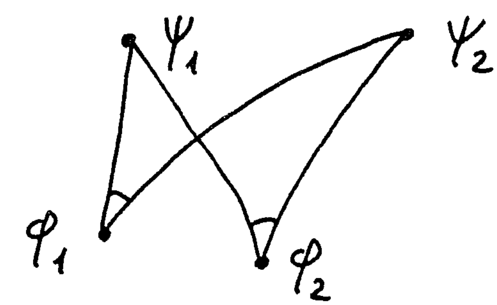
Finite ancestor-graphs

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:



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

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

- Various f
- Various notions of "parents"

The AGO criteria "parents"

Para-parents:

$$\mathrm{id}(\psi_1) + \mathrm{id}(\psi_2) \mod N \quad \text{if paramodulation} \qquad \begin{tikzcd}[column sep=small] \psi_1 \arrow[dr, dash] & & \psi_2 \arrow[dl, dash] \\ & \psi & \end{tikzcd}$$

$$0 \quad \text{otherwise}$$

All-parents:

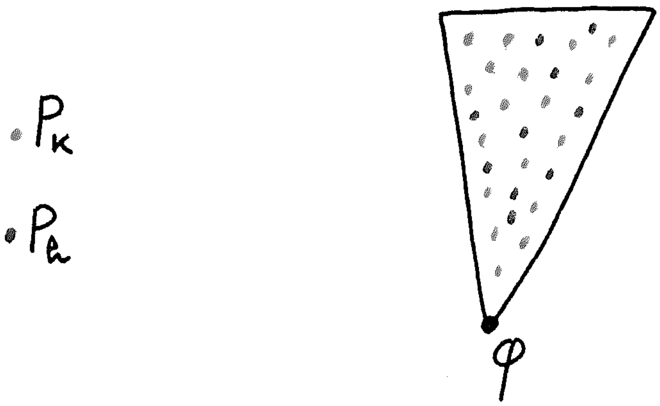
$$\mathrm{id}(\psi_1) + \mathrm{id}(\psi_2) \mod N \quad \text{if paramodulation} \qquad \begin{tikzcd}[column sep=small] \psi_1 \arrow[dr, dash] & & \psi_2 \arrow[dl, dash] \\ & \psi & \end{tikzcd}$$

$$\mathrm{id}(\psi) \mod N \quad \text{if backward-simplification} \qquad \begin{tikzcd} \psi \arrow[d, "*"] \\ \psi' \end{tikzcd}$$

$$0 \quad \text{otherwise}$$

The AGO criterion "majority"

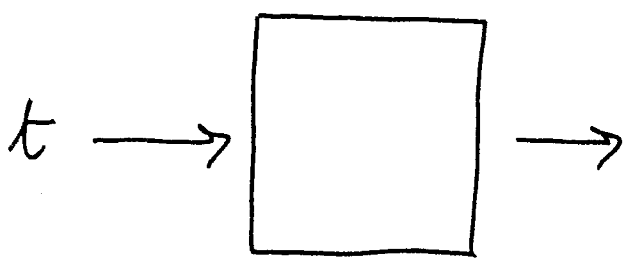
Assign φ to P_k active near φ
(proximity of equations and processes)



ancestor-graph of φ

φ to $P_k \Rightarrow$ increase overlap

Thus: φ to P_h



P_q owns majority in t

Three modes

- Pure distributed search:
  search space subdivided,
  all processes use same search plan.

- Pure multi-search:
  no subdivision,
  every process executes different plan.

- Hybrid:
  search space subdivided,
  different search plans.

Different search plans

1) flag DIVERSE-SEL = 1

P_k uses "given-clause" algorithm if k is odd

P_k uses "pair" algorithm if k is even

2) flag DIVERSE-PICK = 1

PICK-GIVEN-RATIO = x : breadth-first instead of best-first selection every $x+1$ choices

P_k resets it to $x+k$

3) flag HEURISTIC-SEARCH = 1

$$P_k \text{ uses } \begin{cases} h_0 & \text{if } k \bmod 3 = 0 \\ h_1 & \text{if } k \bmod 3 = 1 \\ h_2 & \text{if } k \bmod 3 = 2 \end{cases}$$

Experiments

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

For most experiments with strategy start-m-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.

Moufang identities without cancellation.

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

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

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

Max- weight = 50 for all processes

SGI Onyx shared memory machine:

Sequential time: 149,404 sec or
1 day 17 hr 30' 4" or 41 hr 30' 4"

2 - Peers with majority : 84,0004 sec or
23 hr 33' 26"

speed - up = 1.76
efficiency = 0.88

HP C360 (1G):

Sequential time : 87,007 sec or
24 hr 10' 7"

2 - Peers with all- parents : 34,931 sec or
9 hr 42' 11"

speed - up = 2.49
efficiency = 1.24

Results

Levi Commutator Problem

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

(HP B132L+ with 256M)

Left Moufang identity

Mode	*Search plan*	*EQP0.9d*	*1-Peer*	*2-Peers*	*4-Peers*	*6-Peers*	*8-Peers*
D	given(32)	T	T	598	91	187	40
H	given-h(32)	T	415	230	57	42	**9**
D	pair(32)	3,215	3,277	551	109	51	83
D	4-pair(32)	956	1,068	126	**38**	56	58
D	2-pair(32)	88	130	66	39	109	25
H	2d-diverse-h(32)	88	147	84	75	41	25

Right Moufang identity

Mode	*Search plan*	*EQP0.9d*	*1-Peer*	*2-Peers*	*4-Peers*	*6-Peers*	*8-Peers*
H	given-h(32)	T	437	268	162	100	**28**
D	pair(32)	T	T	865	356	161	105
H	4d-diverse-h(32)	1,558	1,638	75	**32**	27	47

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

Use of parallelism to provide new forms of search for reasoning.

High-performance deduction needs many tools: parallel search by distributed processes is one.

Design / implementation:

- FOL + =
- tools for proof comparison
- more experiments

Theory:

- Semantically-guided distributed deduction

Other current / future work

Strategy analysis of
subgoal-reduction strategies

Application to planning

Application to biology ?

A book on Automated Reasoning