Distributed Automated Deduction

Maria Paola Bonacina

/* Ph.D. Thesis Defense, Dept. of Computer Science,
SUNY Stony Brook, December 4, 1992 */

Outline of the thesis

- Completion procedures as Semidecision procedures

- Distributed automated deduction
  - the Clause - Diffusion methodology
  - the Aquarius prototype

- Directions for future research

Contents of the thesis

1) Completion procedures as semidecision procedures :

- a target-oriented view of derivations by completion procedures

- a new notion of fairness

- a study of contraction and redundancy

- application to completion procedures in equational logic including inductive theorem proving.

Contents of the thesis

2) Distributed automated deduction

a) analysis of the problems in the parallelization of theorem proving (emphasis on contraction-based strategies),

a notion of parallelism at the search level,

b) the Clause-Diffusion methodology,

schemes for distributed global contraction,

distributed allocation of clauses,

Contents of the thesis

b) [cont.]

a study of fairness of distributed derivations,

rules for deleting redundant messages,

distributed subsumption,

c) the Aquarius prototype,

experiments,

discussion.

Outline of the talk

Distributed automated deduction:

- analysis of the problems in the parallelization of theorem proving strategies,

  parallelism at the search level,

- the Clause - Diffusion methodology,

  distributed global contraction,

  Aquarius,

- directions for future research.

Contributions not covered in the talk

Distributed automated deduction:

- distributed allocation of clauses

- message - passing

- fairness of distributed derivations

- inference rules to delete redundant messages

- distributed subsumption

Analysis of the problems in the parallelization of theorem proving strategies

Basic notions on theorem proving strategies

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

I : inference rules

expansion, e.g. resolution, paramodulation,

contraction, e.g. subsumption, simplification.

Forward vs. backward contraction.

Σ : search plan

Derivation :

S_0 : input set of clauses

$S_0 \vdash_{\mathcal{G}} S_1 \vdash_{\mathcal{G}} \ldots \vdash_{\mathcal{G}} S_i \vdash_{\mathcal{G}} \ldots$

Basic notions in theorem proving:

contraction-based strategies

Use eagerly contraction rules to contain the growth of the generated search space;

may feature:

orderings on terms and clauses,

contraction-first search plans,

ordering-based restrictions to expansion.

Classification of strategies for the purpose of parallelization

- Subgoal-reduction strategies:

  e.g. functional programming,
  logic programming,
  PTTP.

- Expansion-oriented strategies:

  expansion,
  possibly forward contraction,
  no backward contraction.

- Contraction-based strategies:

  both forward and backward contraction,
  expansion.

- Subgoal-reduction strategies

$$(S_i ; \varphi_0 ; A_0) \vdash_{\varphi} (S_i ; \varphi_1 ; A_1) \vdash_{\varphi} \ldots \vdash_{\varphi} (S_i ; \varphi_i ; A_i) \vdash_{\varphi} \ldots$$

Static data base.

- Expansion-oriented strategies

$$(S_0 ; N_0) \vdash_{\varphi} (S_1 ; N_1) \vdash_{\varphi} \ldots \vdash_{\varphi} (S_i ; N_i) \vdash_{\varphi} \ldots$$

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

Monotonically increasing data base.

- Contraction-based strategies

$$S_0 \vdash_{\varphi} S_1 \vdash_{\varphi} \ldots \vdash_{\varphi} S_i \vdash_{\varphi} \ldots$$

$$\forall i \quad S_i \subseteq S_{i+1} \quad \text{or} \quad S_i \not\subseteq S_{i+1}$$

Highly dynamic data base.

Granularity of parallelism

	granularity of data	granularity of operations
parallelism at the term level	TERM	SUBTASK OF INFERENCE STEP
parallelism at the clause level	CLAUSE	INFERENCE STEP
parallelism at the search level	SET OF CLAUSES	MANY INFERENCE STEPS

Conflicts

Concurrent processes access the same grain of data :

write - write conflicts between contraction steps,

read - write conflicts between contraction steps,

read - write conflicts between expansion steps and contraction steps.

- Types of parallelism and strategies

	subgoal-reduction	expansion-oriented	contraction-based
parallelism at the term level	✓		
parallelism at the clause level	✓	✓	
parallelism at the search level	✓	✓	✓

- Parallelism at the search level

Concurrent, asynchronous, loosely-coupled deductive processes

develop their own derivations

by working on separate sets of clauses (no conflicts)

and

by exchanging clauses as messages.

Success is reached as soon as one of the processes succeeds.

Distributed environment

- Purely distributed:
  - asynchronous, loosely-coupled processors (nodes),
  - distributed memory,
  - communication by message-passing.

- Mixed shared - distributed:

  it also has a

  shared memory component

  (for simplifiers)

  and thus

  the possibility of combining message-passing with communication through memory.

E.g.: asynchronous multi-processor,

network of computers.

The Clause - Diffusion methodology

Partition the search space among the concurrent deductive processes:

At the clause level:

partition the data base of clauses.

For all clauses ψ, assign ψ to a process P_i:

ψ is a resident of P_i, $\psi \in S^i$.

$$S^0 \cup S^1 \cup \ldots \cup S^{n-1} = S$$

S: global data base

Communication of clauses

Each process P_i may develop a derivation on S^i, but it is not guaranteed to find a proof by using S^i only.

Thus, each process sends its residents to the other processes in form of

"inference messages".

$$(S^i, M^i)$$

S^i: residents at P_i

M^i: inference messages received at P_i

Partition the search space among the concurrent deductive processes:

At the inference level:

expansion inferences:

if $\psi_1 \in S^i$ and $\psi_2 \in M^i$

P_i paramodulates ψ_2 into ψ_1 but not vice versa.

It partitions the search space,
it prevents a systematic duplication
of expansion steps,
it applies also to other rules, e.g. resolution,
including non-binary ones, e.g. hyper-
resolution and unit-resulting resolution.

Contraction inferences:

no general subdivision of contraction steps based on ownership of clauses.

In a contraction-based strategy, each process tries to contract as much as possible residents and messages before expansion and communication.

Local contraction (w.r.t. S^i) and global contraction (w.r.t. $S = \bigcup_{i=0}^{n-1} S^i$):

schemes for

distributed global contraction.

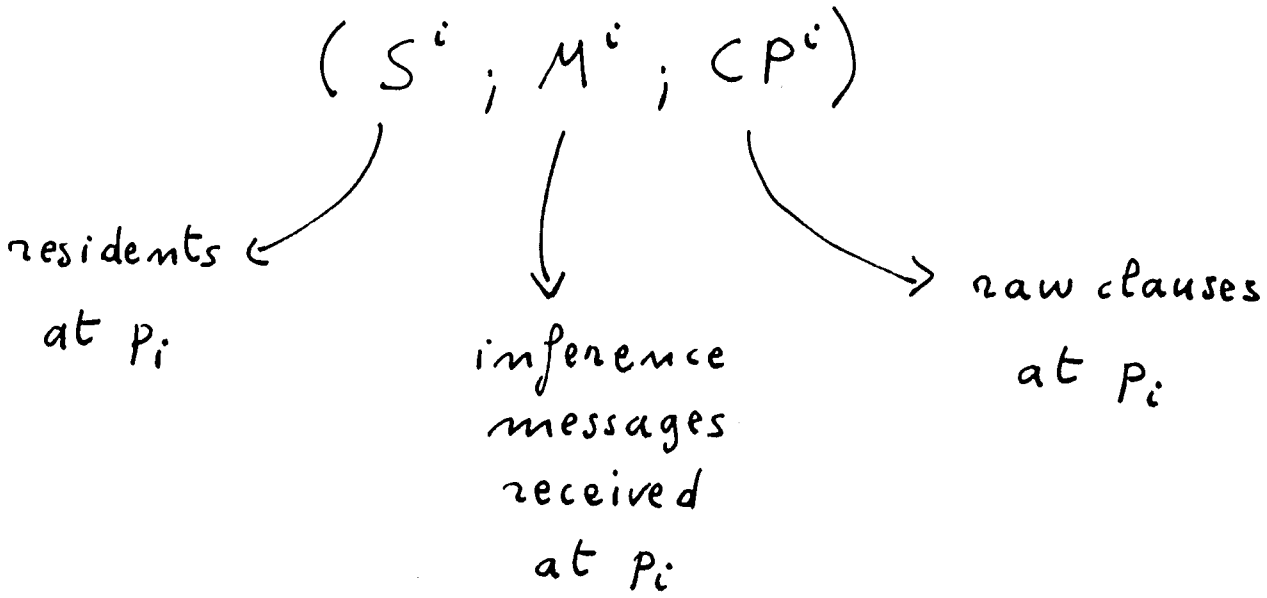
Schemes for distributed global contraction are applied to:

forward global contraction:

on "raw clauses"

backward global contraction:

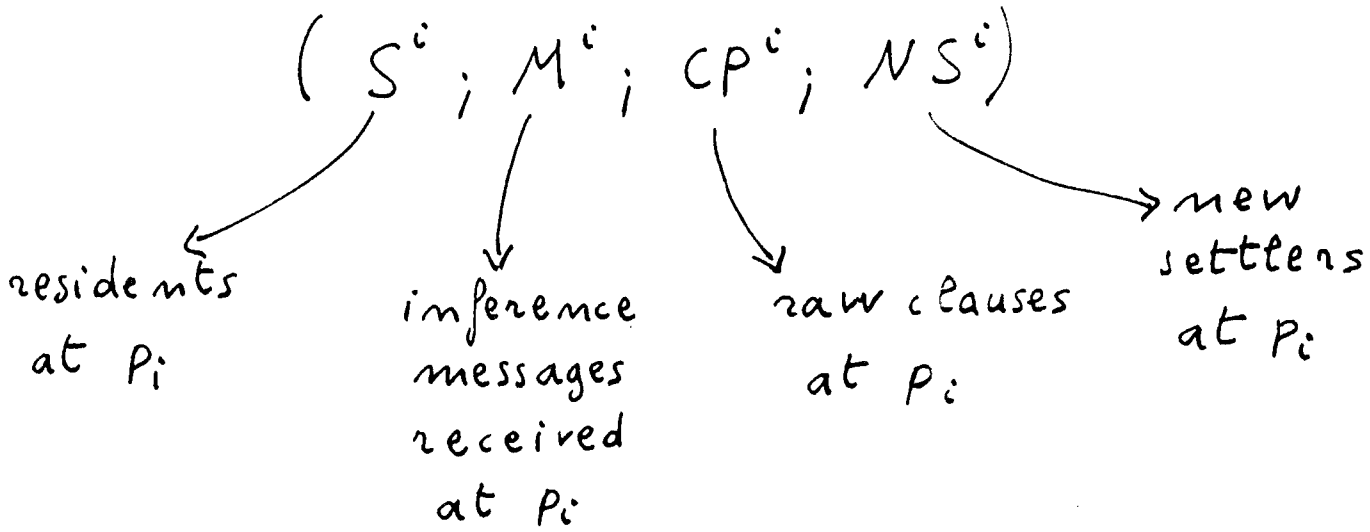
on "residents"



In order to partition the clauses, we need an algorithm for distributed allocation:

after forward contraction a raw clause becomes a "new settler", whose destination is determined by the allocation algorithm

It partitions the search space, while balancing the work-load.



- Policies for the distribution of new settlers

- "Best - fit"
- "Alternate - fit"
- "Half - alternate - fit"
- "First - fit"
- "Alternate - first - fit"

A Clause - Diffusion strategy

features :

- local expansion between residents and between residents and inference messages ⇒ generate raw clauses,
- local contraction of residents and inference messages,
- global forward contraction of raw clauses,
- global backward contraction of residents,
- allocation of new settlers,
- communication of messages.

A Clause - Diffusion strategy

is specified by

- the set of inference rules,
- the search plan which schedules
  - contraction steps
  - expansion steps
  - communication steps

  at each process,
- the algorithm to allocate new settlers as residents,
- the scheme for distributed global contraction,
- the mechanism for message-passing.

Schemes for distributed global contraction

How to access the global data base ?

- By circulating messages :

  global contraction by travelling.

- By accessing an "image set",

  i.e. an "approximated" version of the global data base

  ("approximated" because not up-to-date w.r.t. $\bigcup_{i=0}^{n-1} S^i$) :

  global contraction at the source.

How to realize

global contraction at the source

- Save received inference messages and form a localized image set SH^i at each node p_i : p_i achieves global contraction as contraction w.r.t. SH^i.

- Save "images" of residents in a shared memory component and form a global image set SH : each p_i achieves global contraction as contraction w.r.t. SH.

Maintenance of image sets with respect to contraction

- "Direct contraction"
- "No contraction"
- "Direct update"
- "Delayed update by garbage collection"
- "Update by inference messages"

- Approaches to global contraction

- purely shared
  (backward contraction bottleneck)

- purely distributed, communication-oriented:
  global contraction by travelling

- purely distributed, duplication-oriented:
  global contraction at the source by
  localized image sets

- mixed shared - distributed:
  global contraction at the source by
  global image set in shared memory.

Aquarius

The Aquarius prototype

A distributed theorem prover for FOL+=, written in C and PCN for a network of work stations.

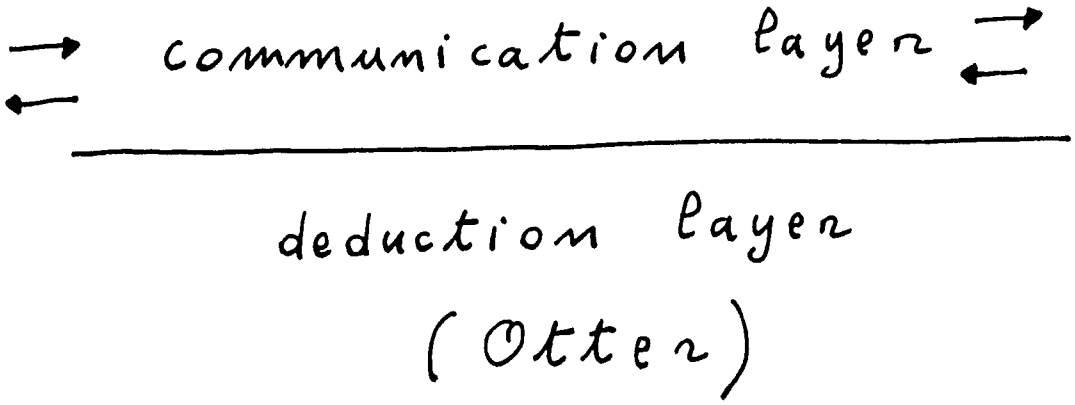
It implements several Clause-Diffusion strategies, with a variety of inference mechanisms, search plans and allocation policies.

Global contraction is done at the source, by localized image sets, updated by inference messages and by direct contraction.

Communication by a fully connected virtual topology of streams.

The Penguin program

Each deduction process in Aquarius executes the Penguin program, which embeds the Otter theorem prover:



It builds on the philosophy of Otter of allowing the user to tailor a specific strategy by selecting its components at both the inference and control level:

121 options.

Problem	*Aquarius-1*	*Aquarius-2*	*Aquarius-3*
andrews	18.00	25.40	24.39
apabhp	11.86	18.11	14.18
bledsoe	12.29	21.53	23.00
boolean-assoc	18.20	21.05	21.18
cd12	104.18	50.98	47.56
cd13	98.79	45.32	51.07
cd90	3.10	1.55	34.85
cn	5.04	8.63	14.50
ec	3.03	1.96	1.77
grp-div	0.37	0.73	0.58
kbgroup	0.27	0.94	0.89
kb-comm	0.65	0.94	1.92
kb-entropic	0.21	0.24	0.20
kb-x2-r	1.79	2.14	5.50
imp1	6.63	2.64	3.54
imp2	7.25	3.31	7.43
imp3	32.05	17.92	38.89
lifsch	2.14	4.96	3.22
luka5	844.20	299.24	1079.45
mission	0.33	0.91	1.04
mv	7.24	6.55	3.60
pigeon	8.21	7.66	8.14
robbins	0.68	1.16	1.16
robbins2	21.62	22.91	24.12
salt	3.89	4.45	5.49
sam-hyp	6.35	5.40	3.90
steam	1.01	1.59	0.57
str-bws	2.32	2.29	2.38
subgroup	15.55	9.36	17.40
tba-1	0.23	0.29	0.86
tba-gg	0.55	0.91	0.84
x2-quant	0.25	0.44	0.97
w-sk	3.50	3.52	3.34

Table 6.1: Experiments with Aquarius (all times are expressed in seconds).

Experiments with Aquarius

The results show irregularities and unstable behaviour.

Communication seems to be too slow.

Limits of the duplication-oriented approach ?

Lack of informed heuristics for the allocation of clauses.

Discussion

- Focus on contraction-based strategies:
  very successful, especially on large problems.

- Parallelism at the search level:
  no synchronization.

- Backward contraction:
  Key role in contraction-based strategies,
  but makes parallelization more difficult.

- "Blind" partition of the search space in
  contraction-based strategies.

Directions for future research

- Development and fine-tuning of Aquarius.

- Quantitative analysis of the performances to identify the difficulties:
  - latency and throughput of communication
  - work - load
  - memory occupancy ...

- Design of heuristics for the allocation of clauses and in general to partition a search space.

- Study of parallel search plans.

Message passing

Resident $\psi \in S^i$: $\langle \psi, a, t \rangle$

Sent as inference message : $\langle \psi, \rho_i, a, t \rangle$

- Routing / broadcasting

  e.g.

  .. on a hypercube

  .. on a fully connected (virtual) topology

  .. on a ring

  Forward reduced messages ?

- Communication through shared memory.

Deletion of redundant messages

Discard Message :

P_k : sender

P_i : receiver $(i \neq k)$

At P_i :

$$\frac{\langle \psi_1, P_k, a, x \rangle \quad , \quad \langle \psi_2, P_k, a, y \rangle}{\langle \psi_2, P_k, a, y \rangle}$$

if

either $\psi_1 \succ \psi_2$

or $\psi_2 \not\succ \psi_1$ and $y \geq x$.

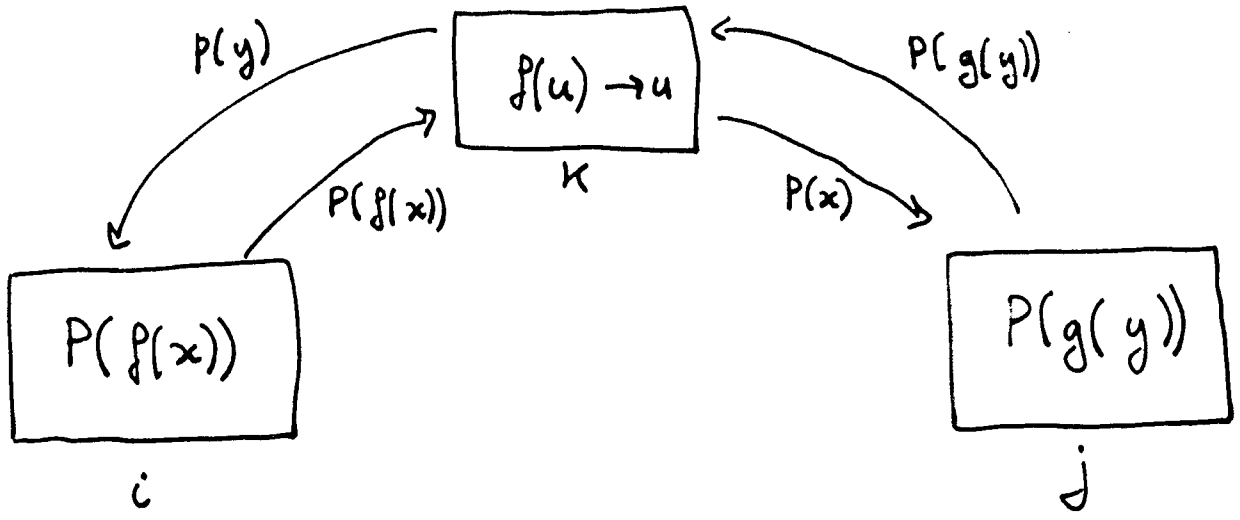
Distributed subsumption

- The problem with subsumption of variants and fairness.

- The problem with subsumption and monotonicity ($S_i \vdash S_{i+1}$, $Th(S_i) \subseteq Th(S_{i+1})$):

  .. subsumption of variants

  .. proper subsumption:



?

C

Sequential subsumption revisited

Replacement subsumption:

$$\frac{\varphi_1, \varphi_2}{\varphi_1, \varphi_1'} \qquad \varphi_2 \geqslant \varphi_1 \;,\; \varphi_1' \doteq \varphi_1$$

Variant subsumption:

$$\frac{\varphi_1, \varphi_2}{\varphi_1} \qquad \varphi_1 \doteq \varphi_2 \qquad (t_2 > t_1)$$

Proper subsumption:

composition of R-subsumption and V-subsumption.

Distributed subsumption

Distributed variant-subsumption:

$\langle \varphi_1, a, t \rangle \in S^\kappa \qquad ds(\varphi_1, \kappa) = (t, \kappa, -)$

$\langle \varphi_1, p_i, a, x \rangle \in M^\kappa \qquad ds(\varphi, \kappa) = (x, i, y)$

$\langle \varphi, s \rangle \in CP^\kappa \qquad ds(\varphi, \kappa) = (\infty, s, -)$

DV-subsumption:

$$\frac{\varphi_1 \,,\, \varphi_2}{\varphi_1} \qquad \varphi_2 \doteq \varphi_1 \;,\; \varphi_2 \gamma_{ds} \varphi_1$$

Distributed subsumption

Distributed replacement subsumption:

$$\frac{\varphi_1, \varphi_2}{\varphi_1, \varphi_1'} \qquad \varphi_2 \rightarrow \varphi_1 \;,\; \varphi_1 \doteq \varphi_1'$$

at process p_k at time z:

- $\langle \varphi_2, p_k, a, t \rangle \in S^k \Rightarrow \langle \varphi_1', p_k, a, z \rangle \in S^k$

  $\langle \varphi_1, p_i, b, x \rangle \in M^k \Rightarrow \langle \varphi_1, p_i, b, \infty \rangle \in M^k$

- $\langle \varphi_2, p_i, b, x \rangle \in M^k \Rightarrow \langle \varphi_1', p_i, b, z \rangle \in M^k$

- $\langle \varphi_2, s \rangle \in CP^k \Rightarrow \langle \varphi_1', z \rangle \in CP^k$

Distributed proper subsumption:

composition of DV-subsumption and DR-subsumption.

Distributed proper subsumption

P_i	P_j
$P(f(x)) \in S^i$	$P(g(y)) \in S^j$
$P(f(x)) \in S^i$ $P(y) \in M^i$	$P(g(y)) \in S^j$ $P(x) \in M^j$
$P(y') \in S^i$	$P(x') \in S^j$
...	...
$P(y') \in S^i$ $P(x') \in M^i$	$P(x') \in S^j$ $P(y') \in M^j$
$P(y')$	