

# Distributed Local Resolution of Boolean Equation Systems

**C. Joubert** R. Mateescu

INRIA Rhône-Alpes / VASY

<http://www.inrialpes.fr/vasy>

PDP'2005 (Lugano, Switzerland)

February 9-11



# Outline

## 1. Boolean Equation Systems (BES)

- Local Resolution of BES
- Related Work

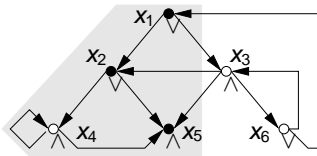
## 2. Distributed Local Resolution of BES

- Distributed Algorithm DSOLVE
- Implementation and Experiments



## BESs, Boolean Graphs, and Local Sequential Resolution

$$\left\{ \begin{array}{l} x_1 \stackrel{\mu}{=} x_2 \vee x_3 \\ x_2 \stackrel{\mu}{=} x_4 \vee x_5 \\ x_3 \stackrel{\mu}{=} x_2 \wedge x_5 \wedge x_6 \\ x_4 \stackrel{\mu}{=} x_4 \wedge x_5 \\ x_5 \stackrel{\mu}{=} \top \\ x_6 \stackrel{\mu}{=} x_3 \vee x_1 \end{array} \right.$$



- ▶ **BES**  $\{x_i \stackrel{\sigma}{=} op_i X_i\}_{1 \leq i \leq n}$ 
  - Set of boolean fixed point equations
  - Pure disjunctive or conjunctive formulas
  - Absence of negations to ensure monotonicity of least fixed point
  
- ▶ **Boolean graph**  $G = (V, E, L)$  associated to a BES
  - $V = \{x_1, \dots, x_n\}$
  - $E = \{(x_i, x_j) \mid x_j \in X_i\}$
  - $L : V \rightarrow \{\vee, \wedge\}, L(x_i) = op_i$

# BES Application Areas

### Verification of concurrent systems

- ▶ Equivalence checking [Andersen-Vergauwen-95]
- ▶ Model-checking [Andersen-94]
- ▶ Partial order reduction [Pace-Lang-Mateescu-03]

### Propositional logic programming

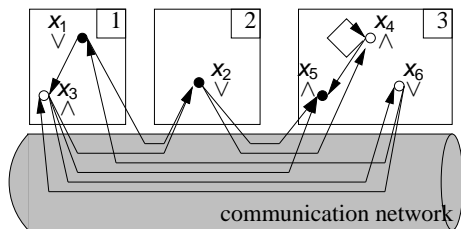
- ▶ Horn clauses satisfiability [Liu-Smolka-98]



## Motivation for Distributed BES Resolution

### 1 Limitation of Sequential BES Resolution Methods

- Memory
  - ▶ BES with more than  $10^8$  variables to solve (current sequential machines swap around  $10^7$  variables)
- Time
  - ▶ Traversals of very large BES (the larger is the BES, the more resolution tasks will have the process)



### 2 Reasons for Distribution

- Running faster with few memory used per machine
- Regular problem prone to balanced distribution of tasks and data

## Related Work

### Distributed Local Resolution using Game Graphs

[Bollig-Leucker-Weber-02]

- ▶ Algorithm specialized for model checking properties expressed in a fragment of the modal  $\mu$ -calculus
- ▶ Operates on game graphs ( $\Rightarrow$  **Alternative representation to boolean graphs**)
  - Vertices = configurations in two-player games
  - Edges = moves
- ▶ Effective speedups obtained for model checking



## The DSOLVE Algorithm

### Computation model

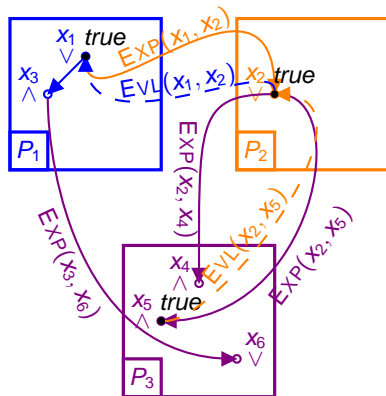
- ▶ Distributed memory architecture (message passing): Now, cluster of PC
- ▶  $P$  SPMD processes and 1 coordinator process
- ▶ Each process solves a subset of boolean variables determined by a static hash function

### Distributed algorithm

- ▶ **Forward exploration** of boolean graph  $(V, E, L)$  starting at a variable of interest  $x \in V$
- ▶ **Backward propagation** of stable (computed) variables
- ▶ **Distribution** (communication) of variables through remote dependencies
- ▶ **Termination detection** when  $x$  stable or boolean graph entirely solved



## Execution of DSOLVE on previous BES example



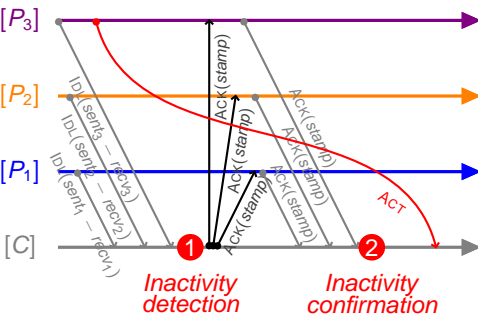
- 1 Initialization (variable of interest  $x_1$ )
- 2 Local expansion and remote expansion (message  $EXP$ )
- 3 Conjunctive variable without successor (i.e., constant **true**)
- 4 Local and remote (message  $EVL$ ) back-propagation of stabilized (i.e., computed) variables
- 5 If variable of interest stabilizes, then resolution terminates



## Distributed Termination Detection Algorithm

## Principle

- ▶ Two waves of global inactivity detection between the coordinator and the resolution processes



## Inactivity detection and confirmation

- 1  $\sum_{i=1}^P \text{inactive}_i = P \wedge \sum_{i=0}^P \text{sent}_i - \text{recv}_i = 0$
- 2  $\text{nb}(\text{ACK}(\text{stamp})) = P \wedge \text{stamp} = \text{current\_stamp}$

## Complexity Results

**For a boolean graph  $(V, E, L)$  and  $P$  running processes:**

- ▶ Worst-case **time** complexity =  $O(|V| + |E|)$ 
  - 2 intertwined graph traversals (forward and backward)
- ▶ Worst-case **memory** complexity =  $O(|V| + |E|)$ 
  - Dependencies stored during graph exploration
- ▶ Worst-case **message** complexity =  $O(2 \cdot |E| \cdot (P - 1)/P)$ 
  - 2 messages (expansion and stabilization) at most exchanged by edges
- ▶ Distributed **termination** detection =  $O(|E| \cdot 3 \cdot P)$ 
  - 2 waves with  $3 \cdot P$  messages at most exchanged by edge



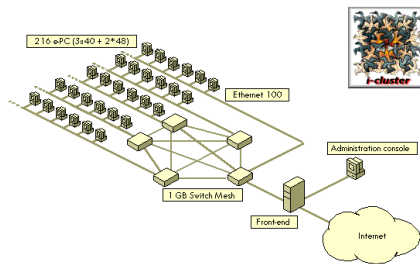
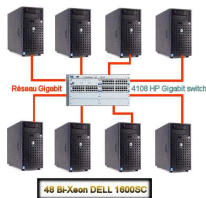
## Distributed Software Architecture

### **DSOLVE (10,000 lines of C code) is based on:**

- ▶ Prototype generic communication library (4000 lines of C code) enabling communication through TCP/IP sockets
- ▶ Generic OPEN/CÆSAR environment for on-the-fly graph exploration [Garavel-98], part of the CADP verification toolbox ([www.inrialpes.fr/vasy/cadp](http://www.inrialpes.fr/vasy/cadp))
- ▶ Generic boolean resolution API given by the library CÆSAR\_SOLVE [Mateescu-03]
- ▶ Available under SOLARIS, LINUX, WINDOWS and MACOS operating systems



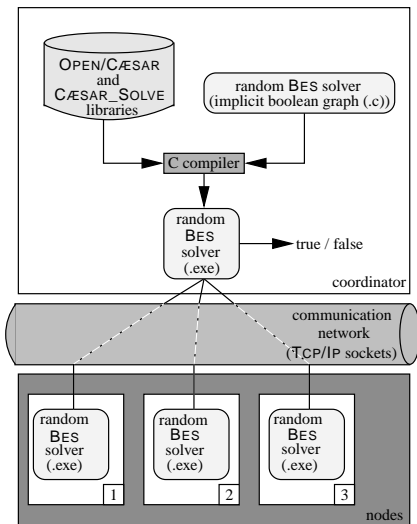
## Platform Architecture



**Experiments performed on homogeneous parallel architectures (Now and clusters of PC), among which:**

- ▶ IDPOT  
(<http://idpot.imag.fr>)  
48 Bi-Xeon 2.5 GHz 1.5 Gb
- ▶ ICLUSTER  
(<http://icluster.imag.fr>)  
216 PIII 733 MHz 256 Mb

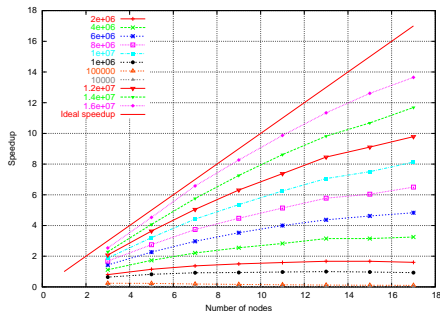
## Experiments



## Random BES Solver

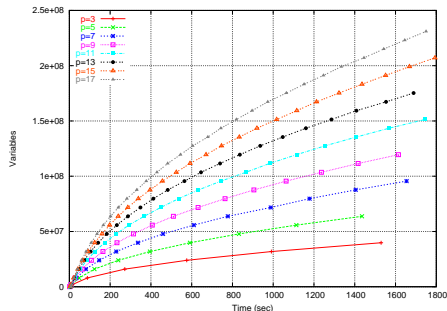
- ▶ Example of small application of DSOLVE (1000 lines of C code)
- ▶ Enables to test the performance of DSOLVE
- ▶ Produces BES (represented by the successor function of their corresponding boolean graph) according to various parameters which vary randomly in a given domain

## Speedup in the Worst Case



- ▶ Worst case class of BES:
  - almost no alternation (2%), hence long path of  $\vee$  (or  $\wedge$ ) variables ended by a constant T (or F) (1%)
  - encountered in model checking (CTL, ACTL, PDL)
  - encountered in equivalence checking involving deterministic graphs
- ▶ Quasi-linear speedup compared to the sequential breadth-first search algorithm of `CÆSAR_SOLVE`

## Scalability



- ▶ Good scalability with the number of machines, as well as with the size of the BES to solve
- ▶ Experimented DSOLVE on more than 80 machines on ICLUSTER
- ▶ Solved  $240 \cdot 10^6$  of variables and  $1 \cdot 10^9$  operators in 28 minutes with 17 machines for a BES with 0% constant and 0% alternation



## Memory and Communication Cost

- ▶ Low memory overhead of distributed resolution compared to memory allocated for data
- ▶ Perfect load balancing achieved by the static hash function
- ▶ High communication cost due to numerous cross-dependencies  $((P - 1)/P) \cdot |E|$
- ▶ Low percentage (0,01%) of termination detection messages over all exchanged messages








# Summary

- ▶ DSOLVE: a new **distributed local BES resolution algorithm**
- ▶ **Generic implementation** within the CADP verification toolbox
- ▶ **Linear worst-case time and memory complexity** DSOLVE algorithm
- ▶ Extensive set of experiments showing **linear speedups and good scalability**
- ▶ Ongoing work
  - Generalizing DSOLVE to BES having several blocks of equations with acyclic inter-block dependencies
  - Developing other applications over DSOLVE, e.g. resolution of Horn clauses, model-checking and test case generation



# For Further Reading

-  D. Bergamini, N. Descoubes, C. Joubert and R. Mateescu.  
BISIMULATOR: A Modular Tool for On-the-Fly Equivalence Checking.  
*TACAS'2005*, LNCS. To appear.
-  C. Joubert and R. Mateescu.  
Distributed On-the-Fly Equivalence Checking.  
*PDMC'2004*, ENTCS. To appear.
-  R. Mateescu.  
A Generic On-the-Fly Solver for Alternation-Free Boolean Equation Systems.  
*TACAS'2003*, LNCS 1443:53–66.

<http://www.inrialpes.fr/vasy>

