DISTRIBUTOR and BCG_MERGE: Tools for Distributed Explicit State Space Generation

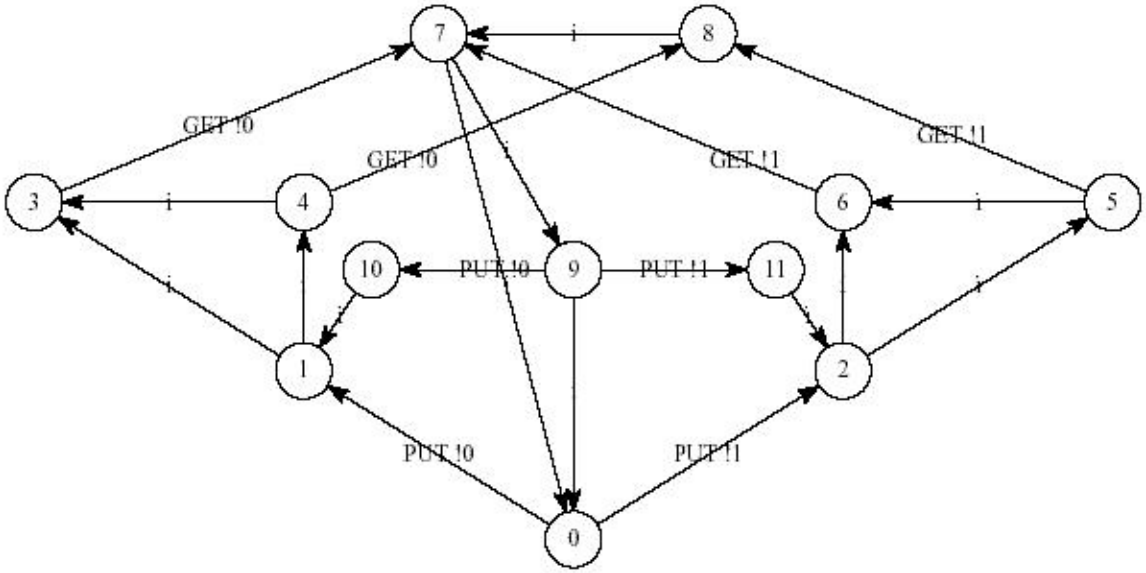
Hubert Garavel, Radu Mateescu, Damien Bergamini, Adrian Curic, Nicolas Descoubes, Christophe Joubert, Irina Smarandache-Sturm, and Gilles Stragier

INRIA Rhône-Alpes / VASY

http://www.inrialpes.fr/vasy

Context and goals

- Explicit state spaces
- Branching-time world (process algebra)
- Labelled Transition Systems



- CADP toolbox (http://www.inrialpes.fr/vasy/cadp)
- Exhaustive state space generation using clusters

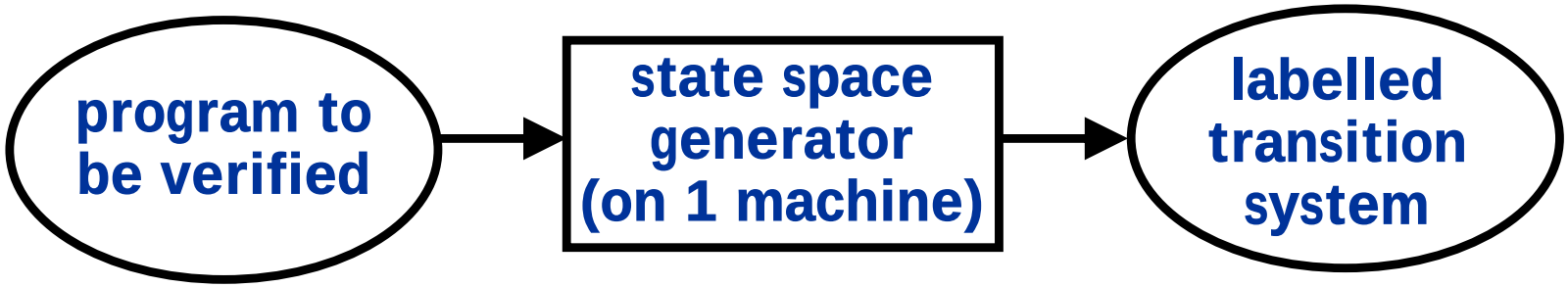
A long-term implementation effort

- DISTRIBUTOR
  - v1: I. Smarandache-Sturm
  - v2: A. Curic and G. Stragier
  - v3: N. Descoubes, C. Joubert, D. Bergamini, H. Garavel
- BCG_MERGE
  - v1: I. Smarandache-Sturm
  - v2: R. Mateescu
  - v3: N. Descoubes, D. Bergamini, H. Garavel
- DISTRIBUTED MONITOR (Tcl/Tk interface)
  - G.Stragier et al.
- File formats improved within the SENVA collaboration (CWI/SEN2 and INRIA/VASY)
  - S. Blom, H. Garavel

Emphasis on "non-standard" issues

- The algorithm for distributed reachability is now standard [Ciardo-Nicols-97]
- Our work focuses on usability-related issues:
  - Generic code libraries for software reuse
  - File formats for describing network resources
  - File formats for storing fragments of state spaces
  - Monitoring protocols: real-time progress information
  - Emergency protocols: node failures, user interrupts, …
  - Graphical user interfaces
  - Proper distribution within CADP, documentation, …

Sequential state space construction



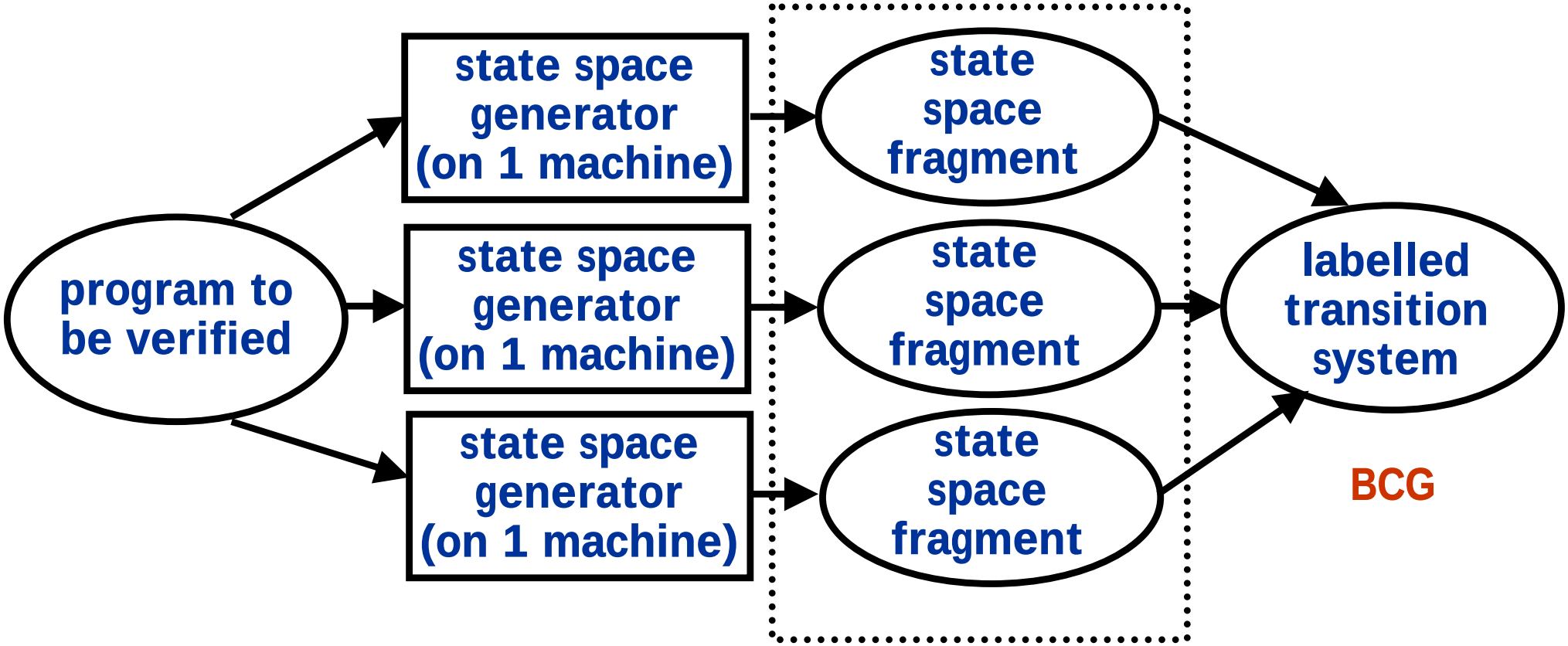
LOTOS, EXP, ...

CAESAR, GENERATOR, ...

BCG *(Binary Coded Graphs)*

- *model checking*
- *equivalence checking*
- *visual checking*

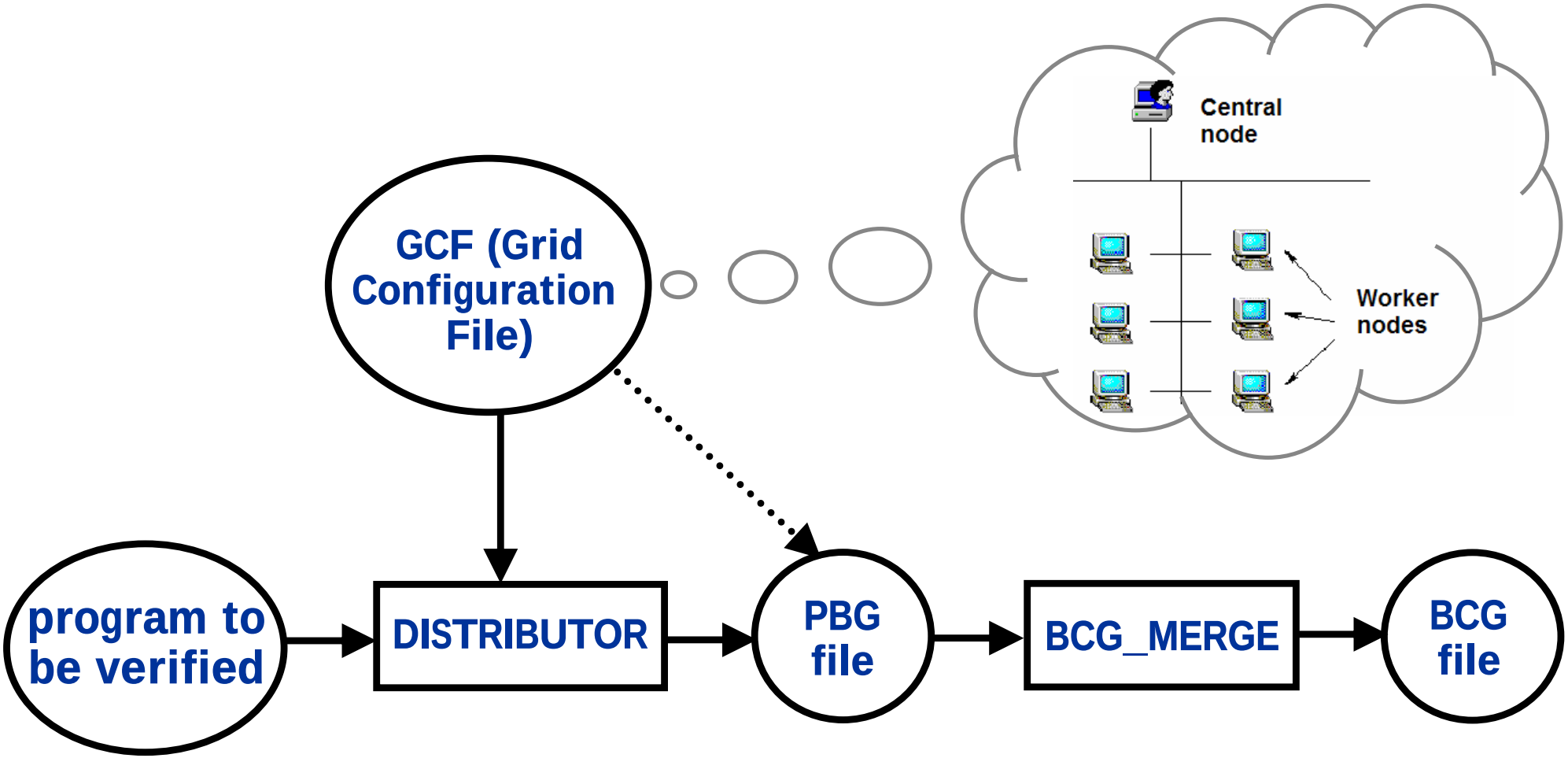
Distributed state space construction



The PBG *(Partitioned BCG Graph)* format

```
PBG 1.0
# PBG format by SENVA team -- http://www.inrialpes.fr/vasy/senva
# created by Distributor (C) INRIA/VASY
# (do not modify this file unless you know what you are doing)
grid: "vasy.gcf"[0]
states: partitioned
edges: incoming
initiator: 5
fragments: 7
1: states: 2667926 fragment: "fragment-1.bcg"[0] log: "1.log"[0]
2: states: 2233636 fragment: "fragment-2.bcg"[0] log: "2.log"[0]
3: states: 1919462 fragment: "fragment-3.bcg"[0] log: "3.log"[0]
4: states: 2653421 fragment: "fragment-4.bcg"[0] log: "4.log"[0]
5: states: 3326293 fragment: "fragment-5.bcg"[0] log: "5.log"[0]
6: states: 2970672 fragment: "fragment-6.bcg"[0] log: "6.log"[0]
7: states: 2666894 fragment: "fragment-7.bcg"[0] log: "7.log"[0]
```

Description of network resources



The GCF (*Grid Configuration File*) format

```
buffer_size = 32768
cadp = /usr/local/cadp
connect_timeout = 10
directory = /home/vasy/distributor
files = graph-*.bcg
hash = 4
port = 8016
rcp = scp
rsh = ssh
user = inria
machine1.domain.org
machine2.domain.org
    user = vasy
machine3.domain.org
    directory = /users/inria/distributor
```

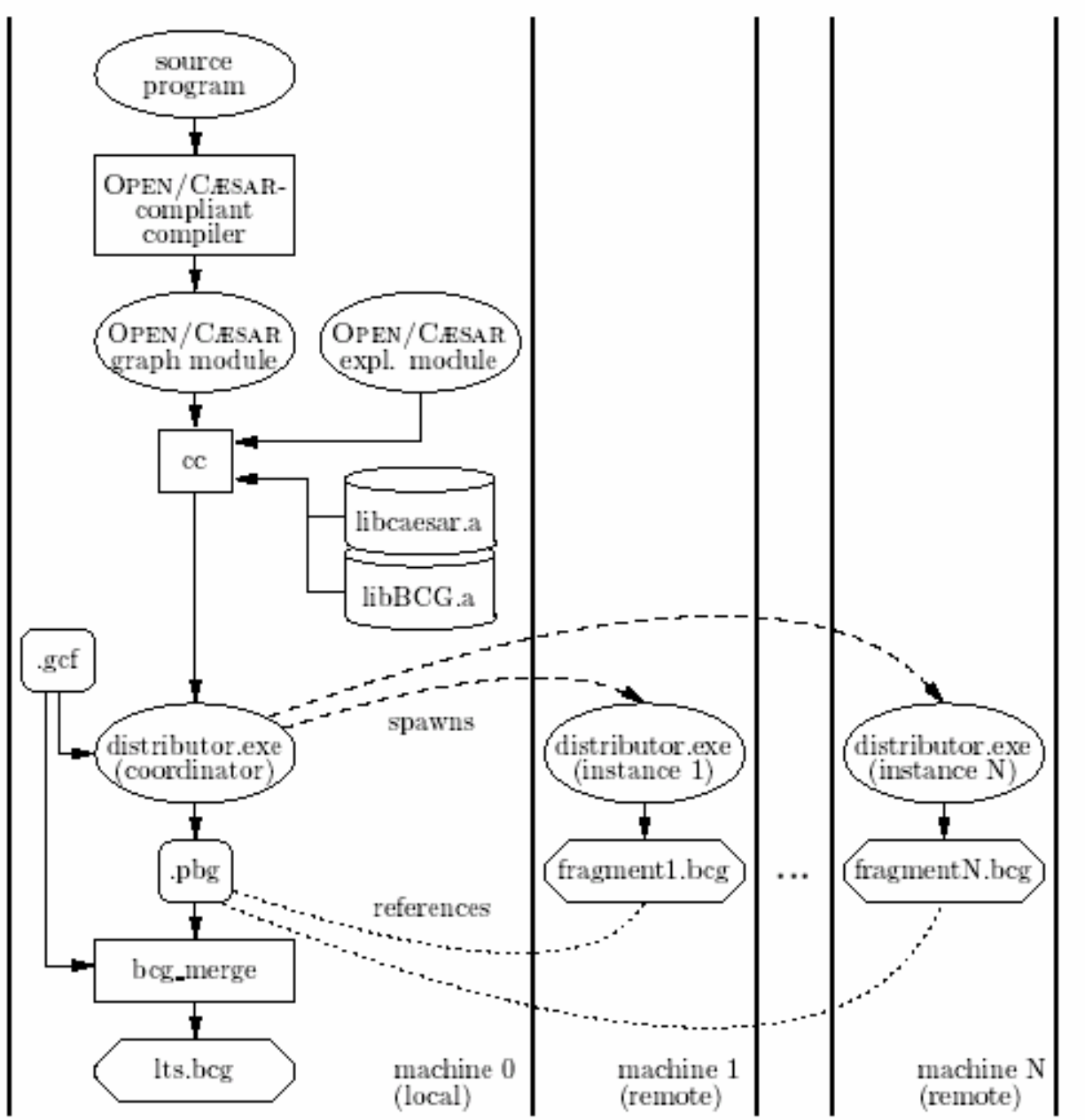
global definitions (applicable to all machines)

list of machines to be used (possibly with local definitions)

The CAESAR_NETWORK_1 library

- A dedicated "verification" middleware
- Minimalistic set of operations:
  - Blocking/non blocking send/receive operations
  - Event waiting
  - Buffered communication
- Based on the most standard technologies:
  - TCP sockets
  - Remote connection using RSH, SSH, or KRSH
  - File transfers using RCP, SCP, KCP
- Ported to Linux, Solaris, Windows, MacOS

Detailed architecture



The demo itself…

- A cache coherency protocol written in LOTOS (Massimo Zendri, Bull)
- 5 instances running on one laptop (1.6 GHz, 512 Mb RAM)
- Configuration file "laptop.gcf":

```
port=8192
localhost
        directory=/tmp/Fragment-1
localhost
        directory=/tmp/Fragment-2
localhost
        directory=/tmp/Fragment-3
localhost
        directory=/tmp/Fragment-4
localhost
        directory=/tmp/Fragment-5
```

Command-line syntax

1) Invocation of Distributor v3:

```
caesar.open cache.lotos distributor -monitor laptop.gcf
result.pbg
```

2) Invocation of Bcg_Merge v3:

```
bcg_merge -monitor result.pbg result.bcg
```

Distributed Monitor

- Real-time monitoring interface
- Built using Tcl/Tk
- Displays numbers of transitions, visited states, remaining states, etc.
- Displays list of labels encountered
- Displays progression status

Additional features of Distributor

On the fly tau-compression:

- Eliminates all tau-cycles (strongly connected components of tau-transitions)
- Preserves branching bisimulation
- Usually fast (linear in the size of the state space)

On the fly tau-confluence:

- Partial order reduction
- Preserves branching bisimulation
- Potentially better reductions than tau-compression
- But potentially slower

Some experimental data

- TU/e Eindhoven (Design and Analysis of Systems)
  - *Dr. Judi Romijn and Stefan Vorstenbosch*
  - Protocols of IEEE P1394.1 draft standard
  - Example 1: 8 M states, 88 M transitions
  - Example 2: 28 M states, 487 M transitions (a few minutes)
- Saarland University (Dependable Systems and Software)
  - *Prof. Holger Hermanns and Sven Johr*
  - Stochastic model of a distributed mutual exclusion algorithm
  - Ex. 1: 44 M states, 87 M transitions
  - Ex. 2: 224 M states, 1.34 G transitions (1.5 hour using 16 procs)
- INRIA/VASY
  - *Dr. Wendelin Serwe*
  - Asynchronous circuit implementing the DES protocol
  - 18 M states, 103 M transitions (50 seconds using 22 Xeons)

Conclusion

- DISTRIBUTOR v3 and BCG_MERGE v3 are now available as part of CADP 2004-*

- One single publication: H. Garavel, R. Mateescu, and I. Smarandache. *Parallel State Space Construction for Model-Checking*. Proc. SPIN'2001.

  Revised version available from http://www.inrialpes.fr/vasy/Publications/Garavel-Mateescu-Smarandache-01.html

Ongoing and future work

Improvements of Distributor and Bcg_Merge

- Support for 64-bits machines and files larger than 3-4 Gbytes
- Support for dynamic data types (pointers)

Rewrite the Caesar_Network_1 library

- Add new functionalities
- Support 64-bit machines
- Support major job schedulers existing on clusters
- Publish its APIs (programming interfaces)
- Distribute it as a "visible" component of CADP

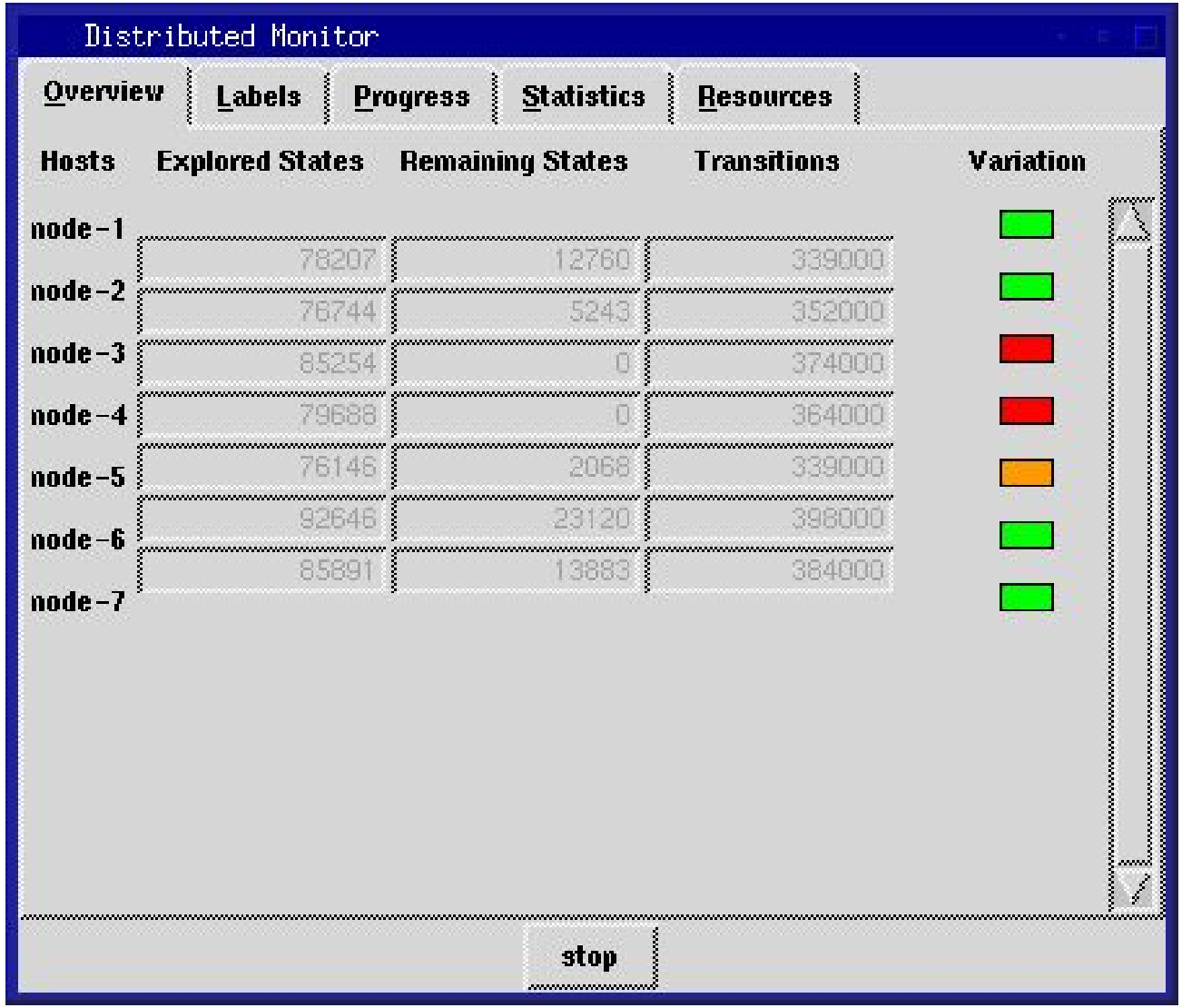
Develop new tools for the PBG format

- Verify PBG models without merging (i.e., without using BCG_MERGE)

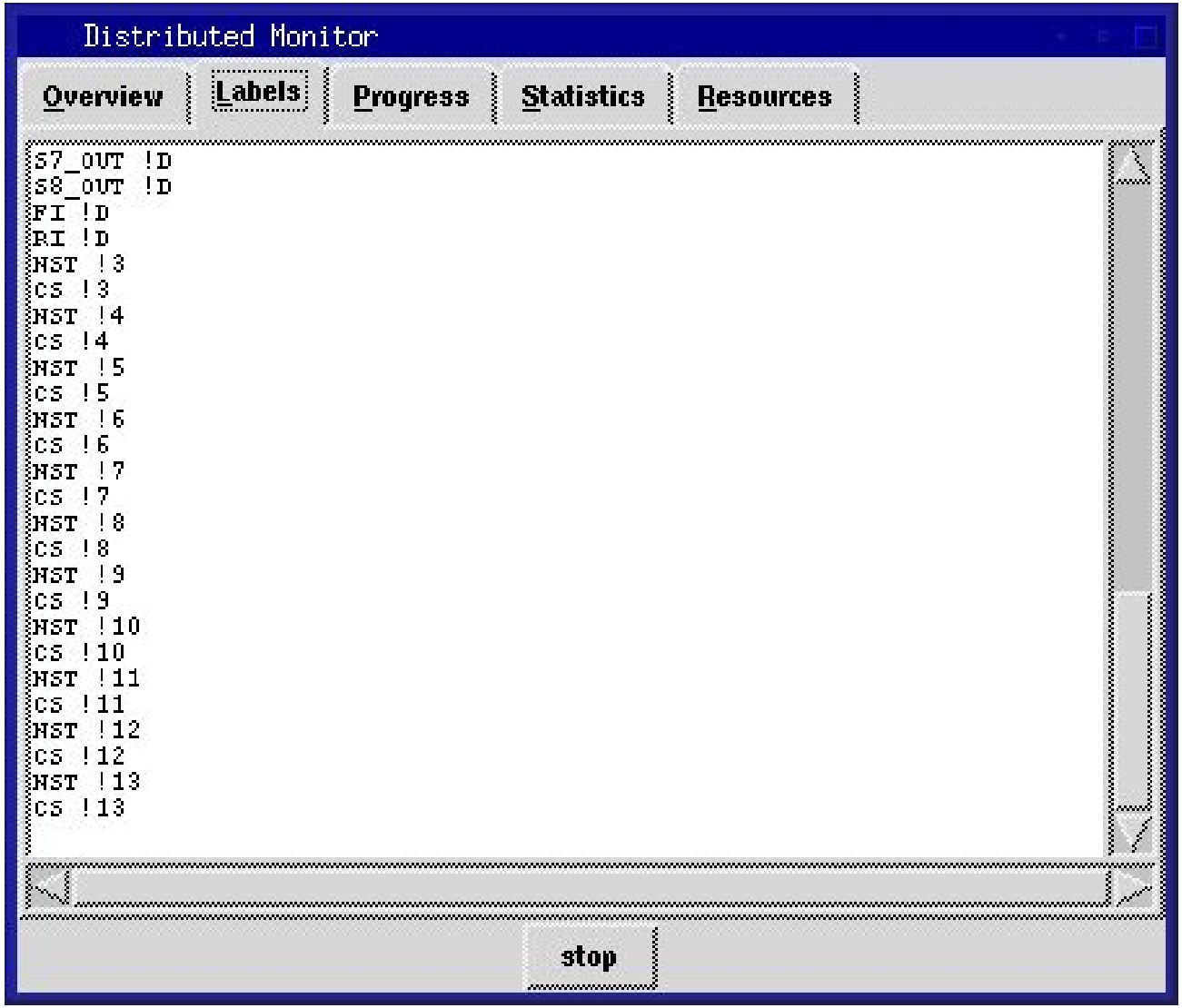
Distributed solver for Boolean Equation Systems

- See [Joubert-Mateescu-04,05,06]
- Also uses the Caesar_Network_1 library

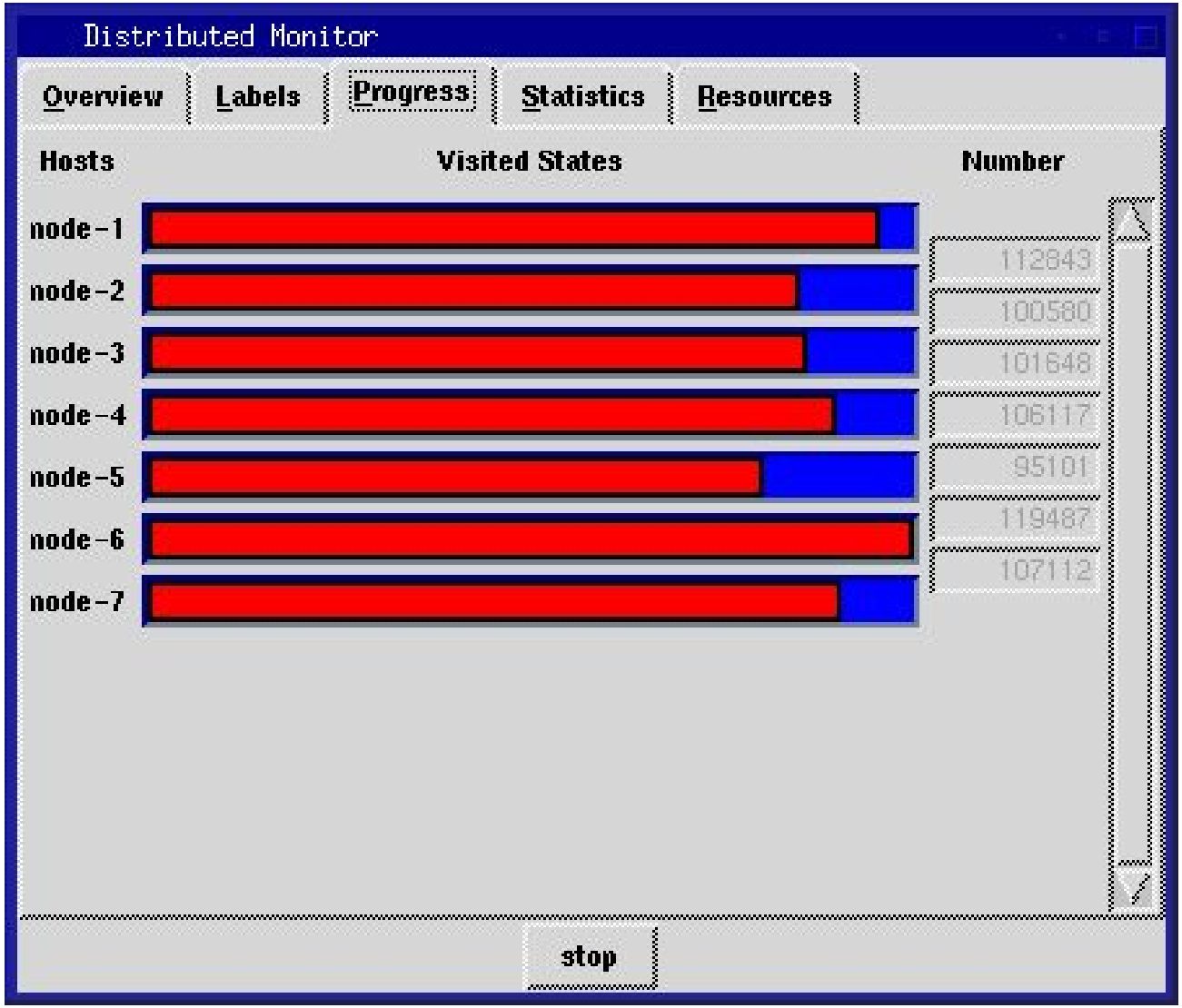
Distributed monitor: “Overview” tab



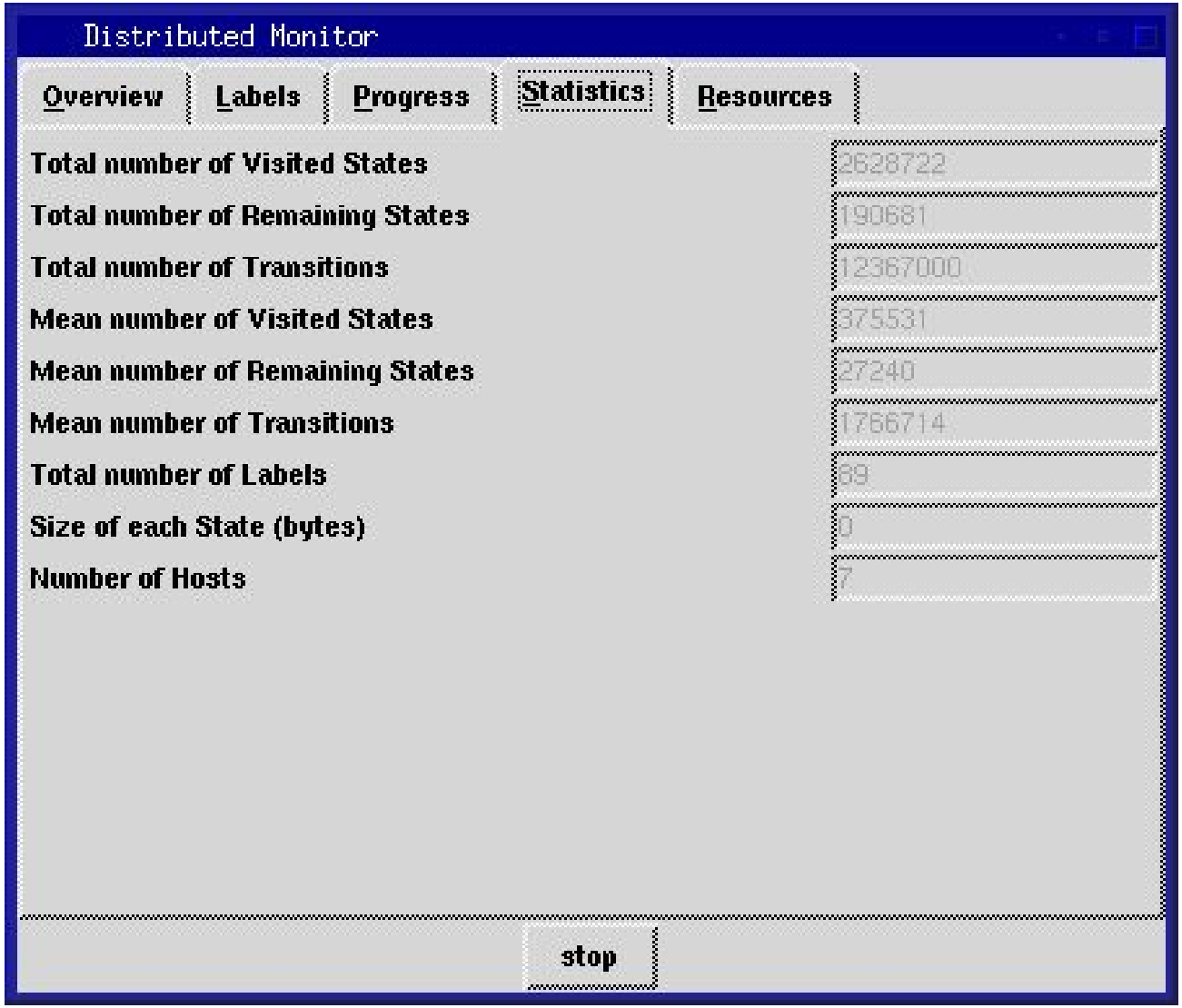
Distributed monitor: “Labels” tab



Distributed monitor: “Progress” tab



Distributed monitor: “Statistics” tab



Distributed monitor: “Resources” tab