EXP.OPEN 2.0
A flexible tool integrating partial order, compositional, and on-the-fly verification methods

Frédéric Lang

INRIA Rhône-Alpes / VASY
655, avenue de l'Europe
F-38330 Montbonnot Saint Martin

EXP.OPEN 2.0

A new tool of the CADP verification toolbox, whose main features are:

- Automata compositions using the operators of several languages (CCS, CSP, LOTOS, μCRL, E-LOTOS, ...)
  - Classical and generalized hide, rename, and cut
  - Classical and generalized parallel composition
  - Synchronization vectors
- Combination of enumerative verification methods
  - Compositional verification
  - On-the-fly verification
  - Partial order reductions
- Connection with PEP and FC2

1. Input language

Examples

```
mcrl behaviour
comm s1|r1 = c1, s2|r2 = c2,
         s3|r3 = c3, s4|r4 = c4,
         s5|r5 = c5, s6|r6 = c6
end comm
hide "c." in
    cut "s.", "r." in
         "snd.bcg" || "rec.bcg" ||
         "m1.bcg" || "m2.bcg"
    end cut
end hide
```

```
csp behaviour
  (
   ("snd.bcg" ||| "rec.bcg")
   [| { s1, s2, r1, r2 } |]
   ("m1.bcg" ||| "m2.bcg")
  ) \ { s1, s2, r1, r2 }
```

```
par
    s1 * s1 * _ * _ -> s1,
    _ * _ * s2 * s2 -> s2,
    _ * r1 * _ * r1 -> r1,
    r2 * _ * r2 * _ -> r2
in "snd.bcg" || "m1.bcg" ||
    "m2.bcg" || "recv.bcg"
end par
```

Labelled Transition System (LTS)

- The semantic model of most process algebras
- Quadruple (S, A, T, s_0), where
  - S and A are the sets of *states* and *labels* (communication events and internal event written *tau*)
  - T is a set of *transitions* between states, labelled by elements of A
  - s_0 is the initial state
- Four file formats available (BCG, Aldébaran, FC2, SEQ)
- Many forms of labels are accepted

Hide, cut, and rename

- Standard notions in process algebras
- EXP.OPEN 2.0 implements μCRL and CCS cut, μCRL, CSP, and LOTOS hide, CCS and CSP rename
- Generalized hide, cut and rename also available
  - Events (gate or label) represented by Posix regexp
  - Hide and cut using negation of a list of events
  - Possibility to define rules in a separate file

Examples

Labels of *B*: PUT(T1), GET(T1), PUT(T2), GET(T2)

- total hide "PUT(T1)" in *B*

  hides "PUT(T1)"

- partial cut all but "T2" in *B*

  cuts all transitions labelled "PUT(T1)" or "GET(T1)"

- gate rename "\(.\)\(.\)\(.\)" $\rightarrow$ "\3\2\1" in *B*

  renames e.g., "PUT(T1)" into "TUP(T1)"

- multiple rename "T" $\rightarrow$ "TT" in *B*

  renames e.g., "PUT(T1)" into "PUTT(TT1)"

Parallel composition

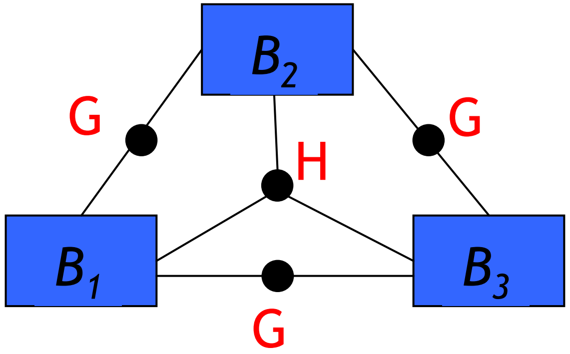
- Binary infix parallel composition operators from CCS, CSP, LOTOS, and μCRL
- Generalized parallel composition operators
  - Parallel composition using synchronization vectors
  - E-LOTOS parallel composition

E-LOTOS parallel composition

Generalization of LOTOS (Garavel & Sighireanu, 1999)

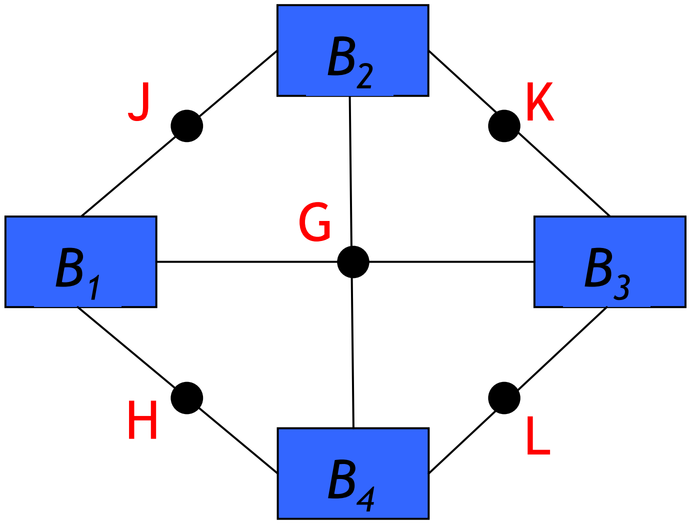
- Forced synchronization on a set of events for *n* concurrent processes
- Relaxed forms of synchronization for particular events
  - *n* among *m* synchronization
  - Interface synchronization

Examples



$\rightarrow$

gate par G#2, H in
B_1
|| B_2
|| B_3
end par



$\rightarrow$

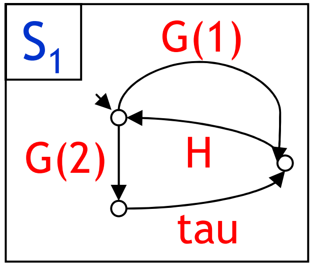
gate par G in
H, J $\rightarrow$ B_1
|| J, K $\rightarrow$ B_2
|| K, L $\rightarrow$ B_3
|| L, H $\rightarrow$ B_4
end par

2. State space exploration using EXP.OPEN 2.0

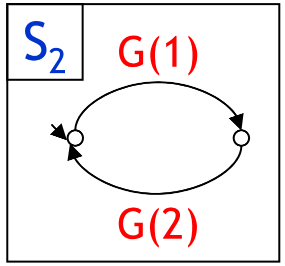
Compilation into an internal form

Every expression is compiled into a vector of LTSs and a set of synchronization vectors

- Allows an homogeneous treatment of expressions
- Example: rename "H" → "K" in

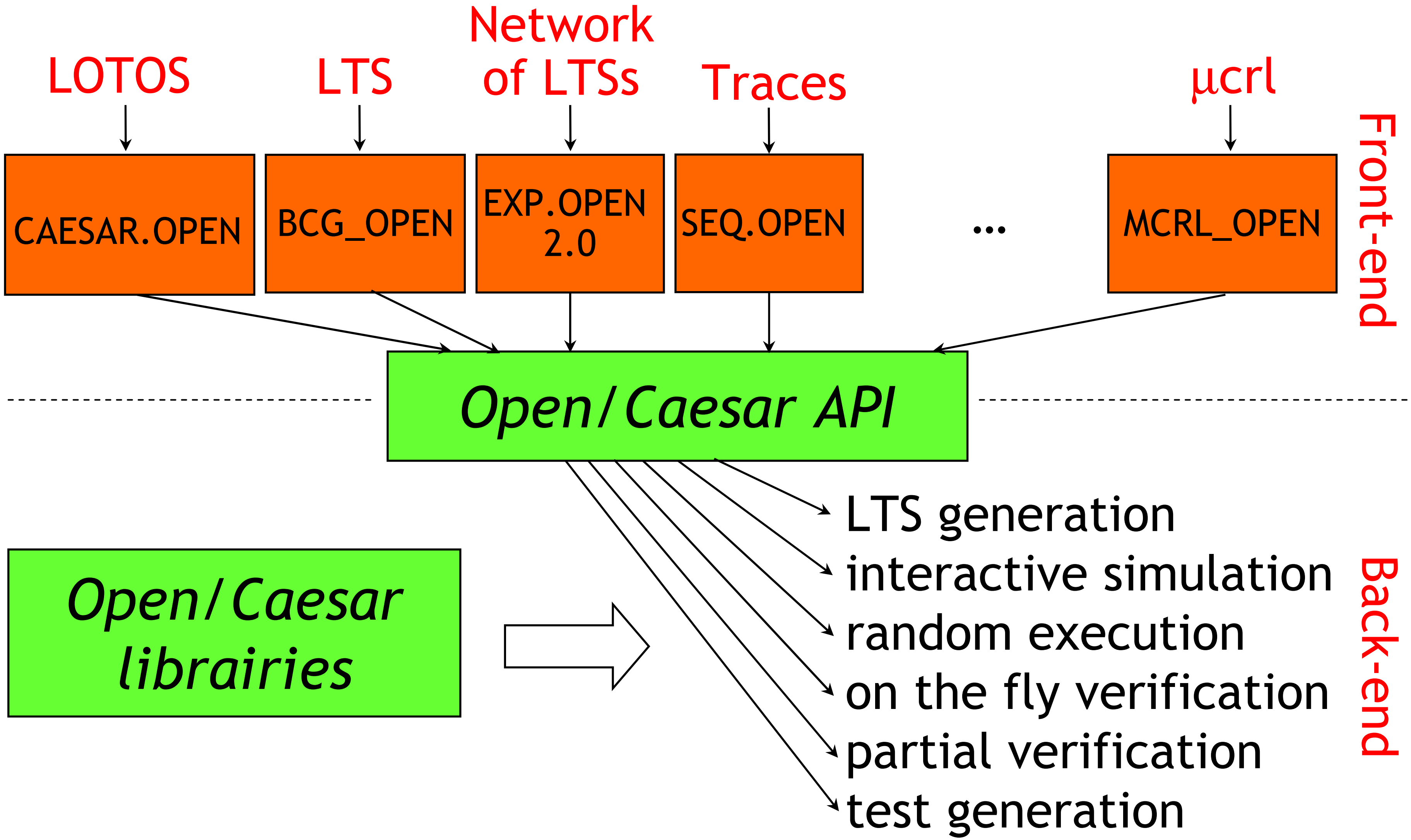


|[G]|



is compiled into (S_1, S_2), { ("tau", _) → "tau",
("H", _) → "K",
("G(1)", "G(1)") → "G(1)",
("G(2)", "G(2)") → "G(2)" }

Integration within OPEN/CAESAR (CADP)



Partial order reductions

- Select a partially ordered execution of a set of transitions to avoid exploring all interleavings
- Three partial order reductions are implemented, preserving different relations
  - Branching bisimulation
  - Stochastic branching bisimulation
  - Deadlocks
- No modification of the verification back-ends

Compositional verification

- Standard approach: Generate, hide, and reduce modulo an equivalence incrementally
  - May avoid explosion of the full LTS
  - Sound as the main equivalences are congruences
  - But limited by possible explosion of an intermediate LTS
- Refined approach: Additionally use interface constraints to generate LTSs
  - Interface = LTS modeling an abstraction of the context
  - Proposed by Graf & Steffen (90) and implemented in the PROJECTOR tool by Krimm & Mounier (97)

Interface constraints generation

- EXP.OPEN allows to generate interface constraints for a process in a known context automatically
- Main advantages:
  - It avoids generation of constraints by hand
  - It is not restricted to a particular language
  - Interface constraints can be built upon any subset of the processes in the context of the process to constrain
  - It improves over other approaches in the case of nondeterministic synchronization
- For LOTOS, the approach is automated within SVL

3. Applications and performances

Various applications

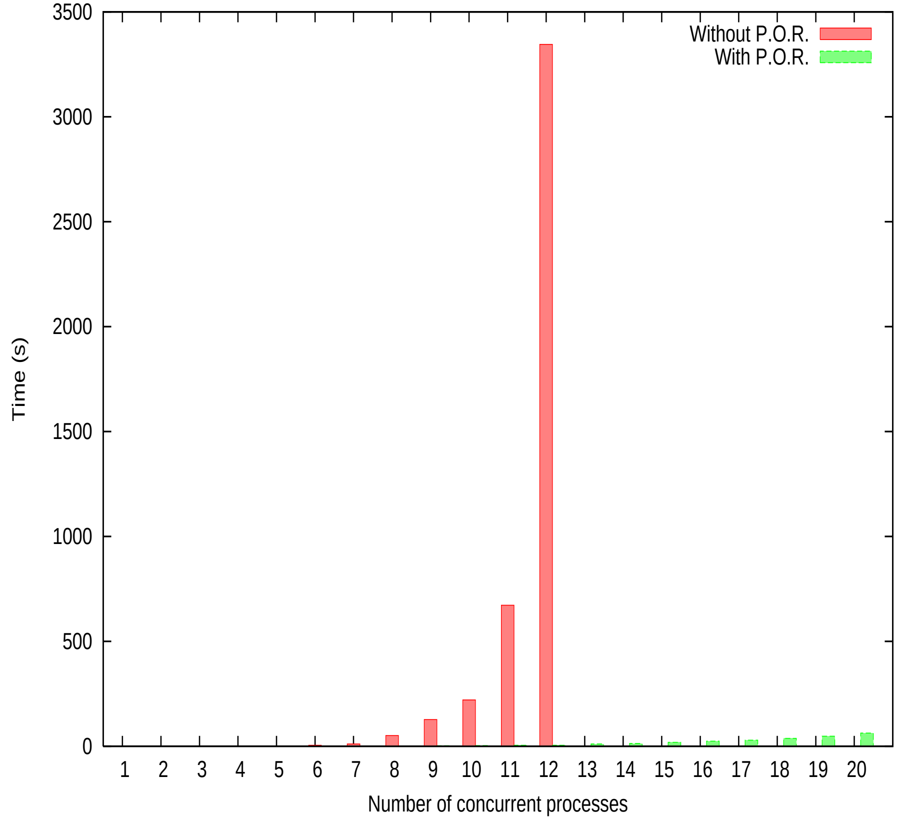
- Verification of Net update protocol (Firewire)
  - By Romijn, Vorstenboch, and Huo (Univ. of Eindhoven)
  - Distributed state space generation
- Performance analysis of a distributed mutual exclusion algorithm
  - By Hermanns and Johr (Saarland University)
  - Branching stochastic partial order reduction and distributed state space generation
- Compositions of hierarchical object components
  - By Barros and Madelaine (INRIA)
  - Hierarchical compositions using synchronization vectors

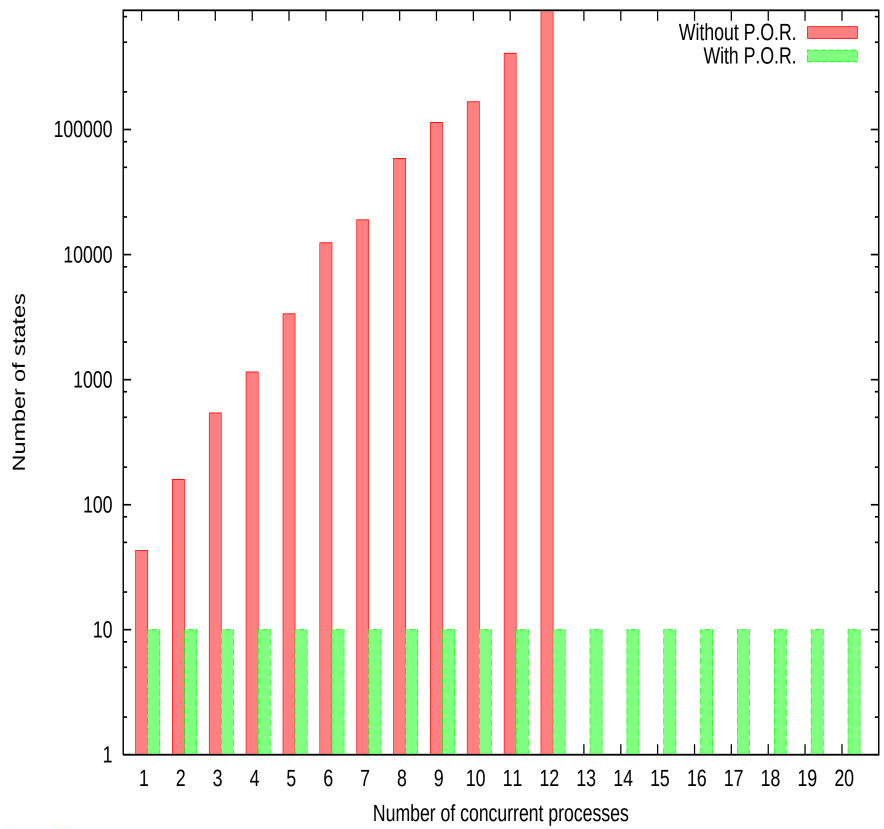
Various applications (continued)

- Several online demos available in CADP
  - Distributed summation algorithm using *m* among *n* synchronization
  - ODP trader using *m* among *n* synchronization
  - Distributed Erathostene's sieve using partial order reduction
  - Compositional verification of Data Encryption Standard (DES)
  - etc.

Performances

Erathostene’s sieve using partial order reduction preserving branching bisimulation

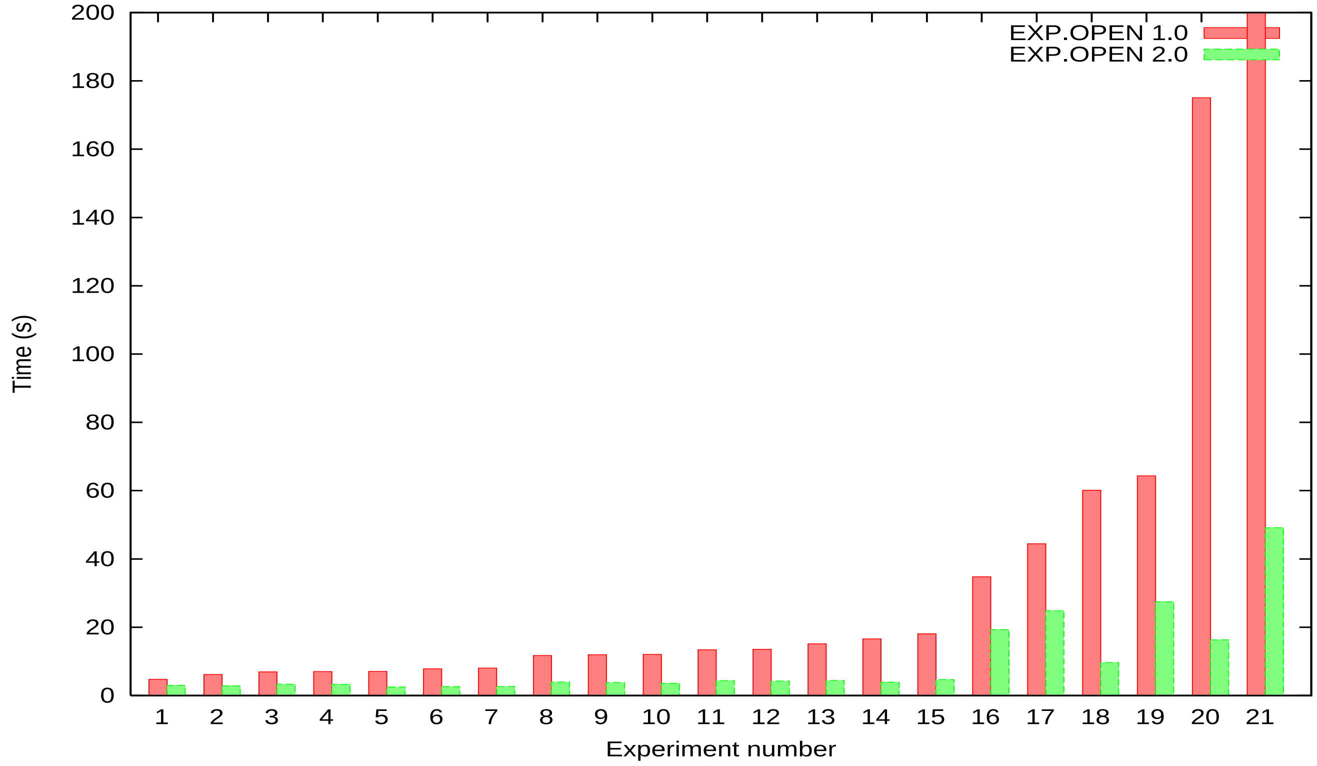




Performances (cont'd)

Significant improvements wrt. version 1.0

- Memory divided by 2
- Time:



Conclusion

- EXP.OPEN 2.0 combines operators
  - Classical and generalized process algebra operators
  - Synchronization vectors
  - First implementation of E-LOTOS parallel composition
- EXP.OPEN 2.0 combines techniques
  - On-the-fly verification and partial order reductions
  - Compositional verification with interface constraints
- More information on CADP and EXP.OPEN 2.0:

http://www.inrialpes.fr/vasy/cadp