Refined interfaces for compositional verification

Frédéric Lang

INRIA Rhône-Alpes

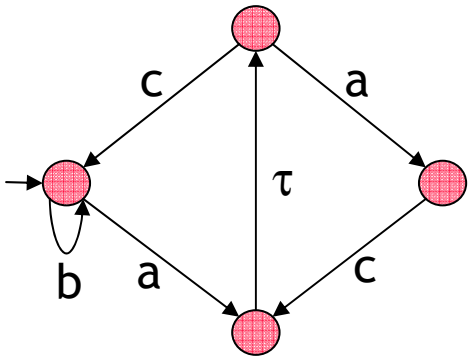
http://www.inrialpes.fr/vasy

Motivation

- Enumerative verification of concurrent systems
  - Parallel composition of asynchronous processes
  - Systematic exploration of the state/transition graph obtained by interleaving and synchronization
- Compositional verification to palliate state explosion
  - Simple: reduce state/transition graphs incrementally
  - Enhanced: use interface constraints to avoid intermediate state explosion
- This talk is about a tool to build interface constraints automatically

State/transition graphs

- Semantic model of action-based processes, also called Labelled Transition System (LTS)
- Transitions between states are labelled by events
  - Synchronizable/observable events
  - Non-synchronizable/hidden event τ

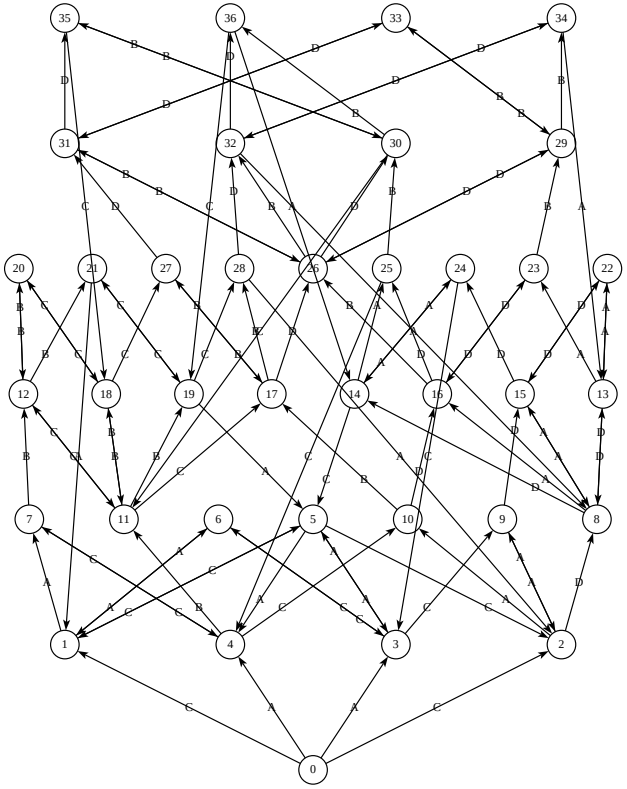


- CADP toolbox allows on-the-fly exploration of state/transition graphs (OPEN/CAESAR)

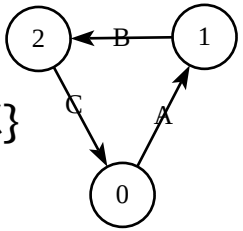
Using interface constraints

- A big graph P can be reduced using *interface constraints*, represented as a graph I and a set of labels A through which P and I interact
- Projection operator $P \rceil|_A I$ (Graf & Steffen, Krimm & Mounier)
  - Computes the sub-graph of P reachable in $P \;||_A I$
  - I can be reduced modulo safety equivalence after hiding all labels outside A
- A similar approach exists for CSP (Cheung & Kramer)
  - Normal parallel composition instead of projection
  - Requires tau elimination and determinization (expensive) in I to ensure *context transparency*

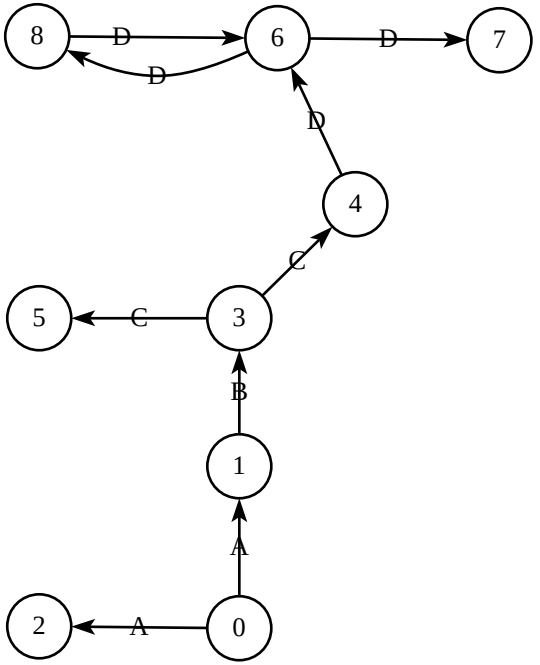
Example of projection



$\lceil\!\lceil_{\{A, B, C\}}$

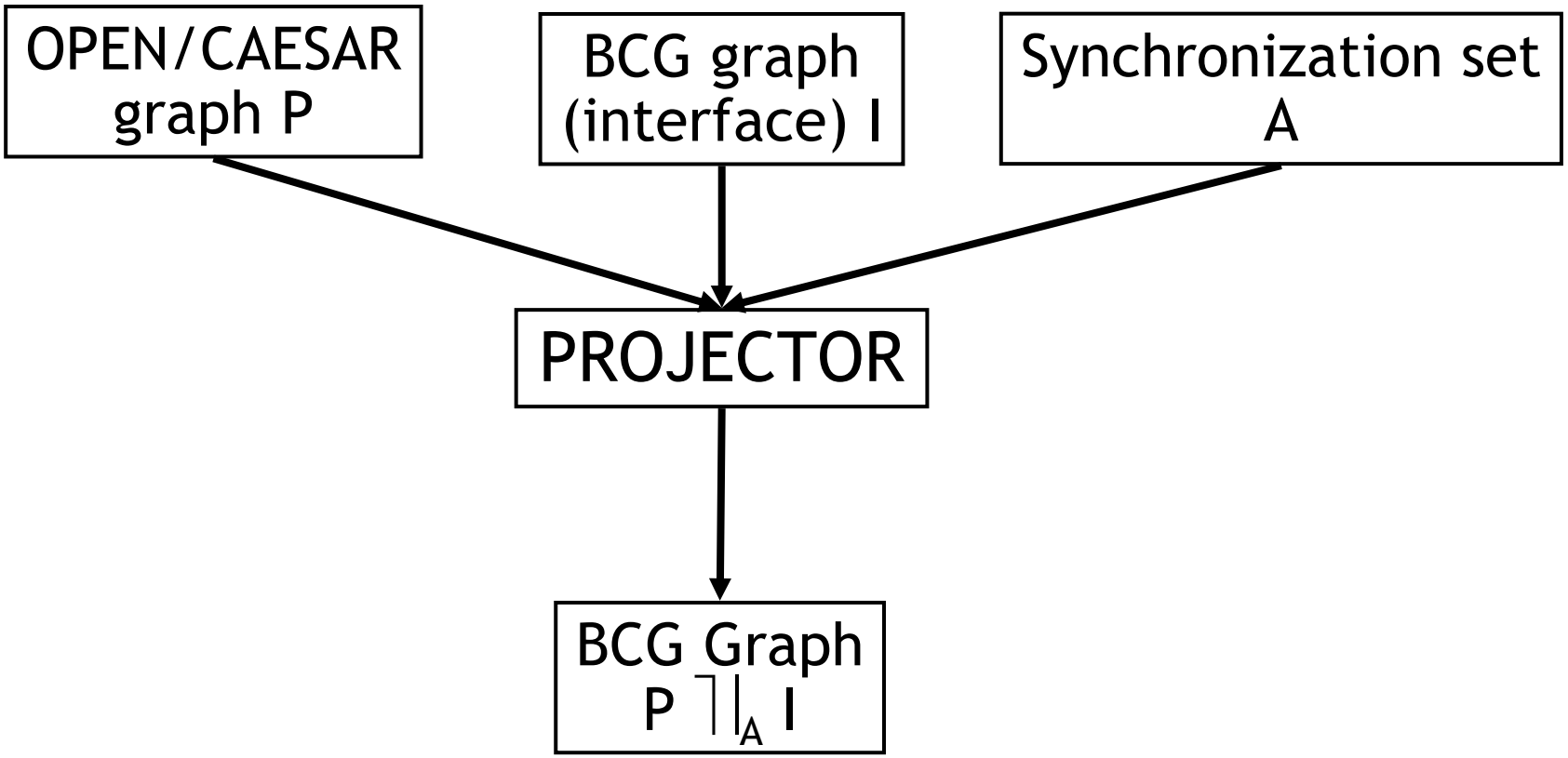


=



The PROJECTOR tool of CADP

Software implementation of projection (Krimm & Mounier 1997)



Computing the interface constraints

- **Solution 1:** User-specified interface
  - The user provides an interface
  - A correct interface is hard to guess
  - But correctness can be checked afterwards
- **Solution 2:** "Exact" interface
  - A correct interface is computed automatically from the environment
  - Krimm & Mounier give an algorithm based on an analysis of the algebraic LOTOS-like expression describing the composition of processes
    - The interface I is a process of the composition
    - The synchronization set A is derived automatically

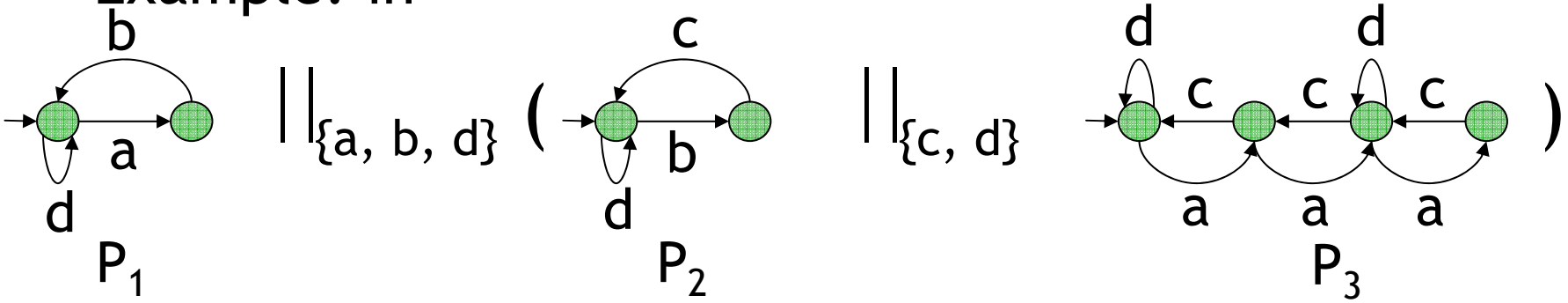
Limitation 1 of K&M algorithm

The method to compute the synchronization set A is specific to LOTOS parallel composition

How can we build exact interfaces in expressions that use different and/or more general operators?

Limitation 2 of K&M algorithm

- It is impossible to compute interface constraints induced by **a combination of** (distant) processes
- Sometimes, only such constraints allow reductions
- Example: in

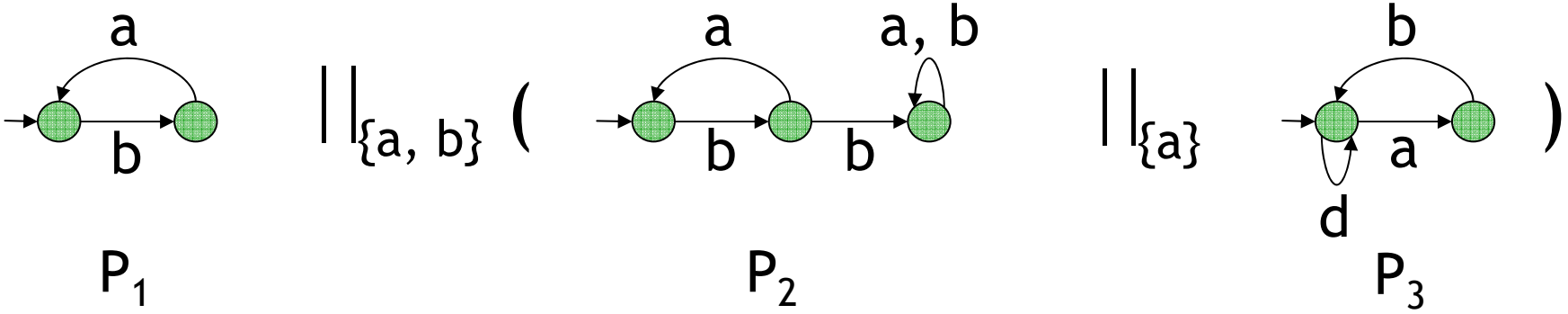


restricting P_3 w.r.t. either P_1 or P_2 yields no reduction: $P_3 \lceil\!|_{\{a, d\}} P_1 = P_3 \lceil\!|_{\{c, d\}} P_2 = P_3$

- Using an interface obtained by combination of P_1 and P_2 (synchronized on b) would yield better reductions

Limitation 3 of K&M algorithm

- Interfaces may be not precise enough when nondeterministic synchronization is involved
- Example: in



Restricting P_2 w.r.t. P_1 yields no reduction:

$P_2 \lceil\!|_{\{a\}} P_1 = P_2$

- However P_1 implies that two successive b actions cannot be reached without an a in between

Refined interfaces

- We propose a new algorithm which solves the limitations of K&M algorithm
- The algorithm works in three phases
  1. Translation of the composition of processes into a general model called "synchronization networks"
  2. Extraction of an "interface network" from the network model
  3. Generation of the interface graph corresponding to the interface network

Phase 1: synchronization networks

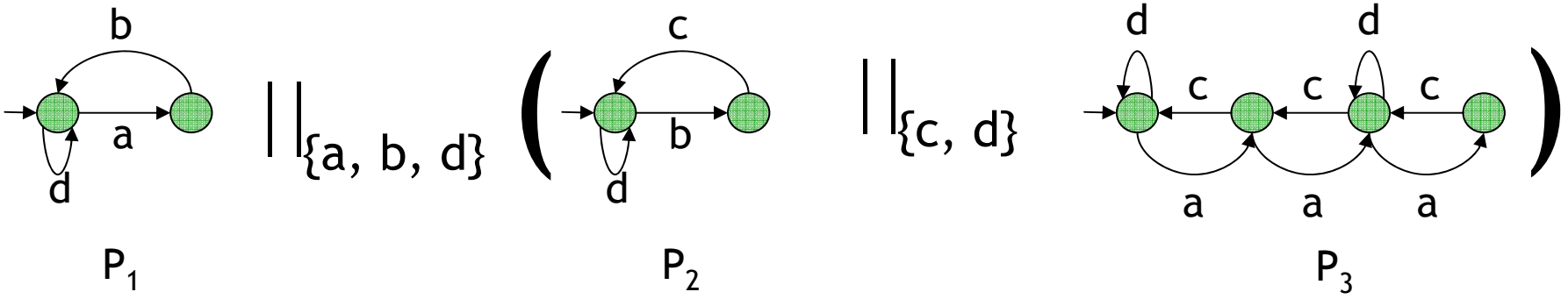
- A general synchronization model
- Synchronization of processes P_1, ..., P_n described by a set of synchronization vectors of the form

$$L_{i,1}, \ldots, L_{i,n} \rightarrow L_i$$

where

  - each $L_{i,j}$ is either a label of P_j or the special symbol • denoting inaction of P_j
  - L_i is the label used in the product graph as the result of the synchronization between the P_j's

Example 1



P_1 P_2 P_3

can be represented by the set of synchronization vectors

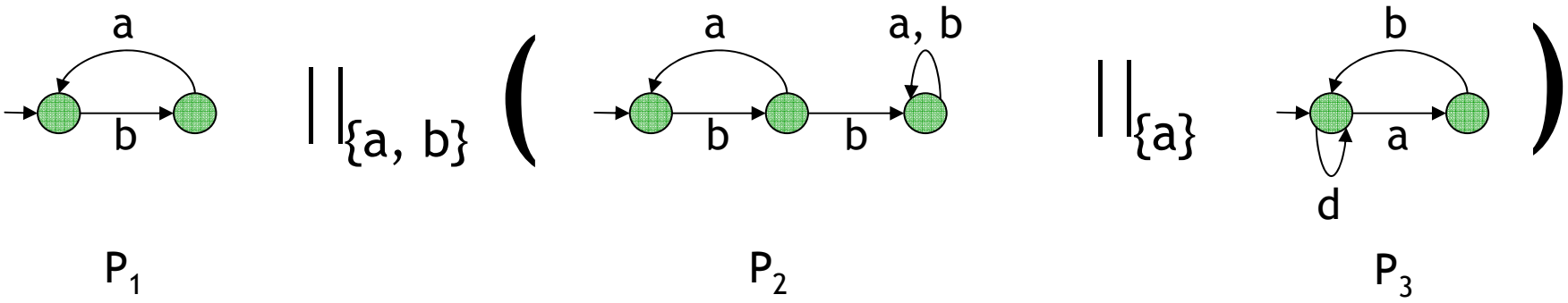
$a, \bullet, a \rightarrow a$

$b, b, \bullet \rightarrow b$

$\bullet, c, c \rightarrow c$

$d, d, d \rightarrow d$

Example 2



can be represented by the set of synchronization vectors

a, a, a → a
b, b, • → b
b, •, b → b
•, •, d → d

} nondeterministic synchronization on b for P_1

Phase 2: Interface network extraction

- Extraction of a network N' representing an abstraction of the environment of a process to be constrained
- Inputs:
  - The synchronization network N of a system P_1, ..., P_n
  - The index i of the process P_i to be constrained
  - The indices j_1, ..., j_m (user-given) of the constraining processes
- Algorithm: for each vector v in N, create in N' a vector

  $v[j_1], \ldots, v[j_m] \rightarrow r$

  where $r = v[i]$ if $v[i] \neq \bullet$ (P_i active in synchronization)

  $r = \tau$ otherwise

Example

- P1, P2, P3 synchronized by the vectors

  a, a, a → a

  b, b, • → b

  b, •, b → b

  •, •, d → d

- The interface network of P2 induced by P1 is:

  a → a

  b → b

  b → τ

  ~~• → τ~~ (This last one can be removed)

Phase 3: interface graph generation

- Generate the graph corresponding to N' (product of P_{j1}, ..., P_{jm})
- Thanks to congruence, P_{j1}, ..., P_{jm} can be reduced modulo safety equivalence beforehand
- Partial order reduction allows to avoid useless interleavings

Using the generated interface

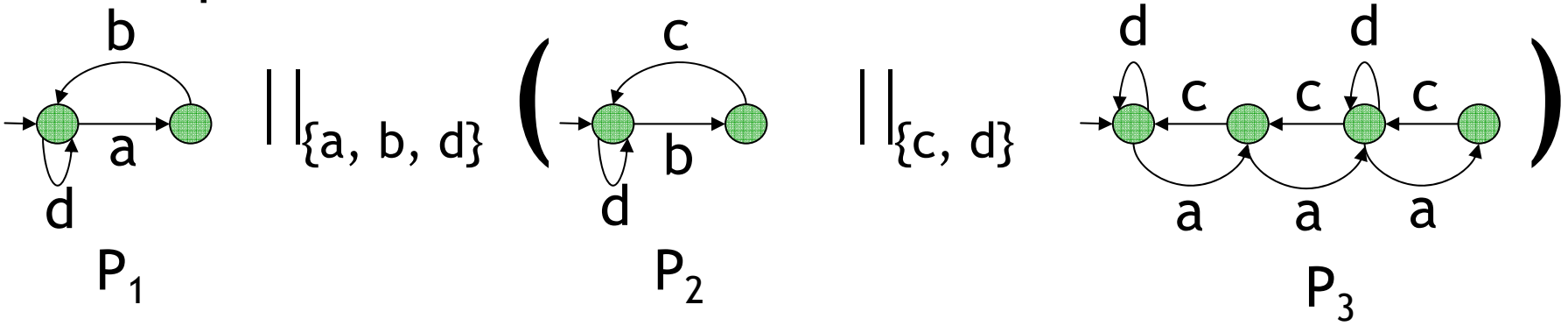
- The (possibly large) graph of P_i can be replaced by (smaller) graph of $P_i \lceil\!\lceil_A I$ where I is an interface obtained by our algorithm
- Formal proof provided in FORTE'2006 paper

Limitation 1 solved

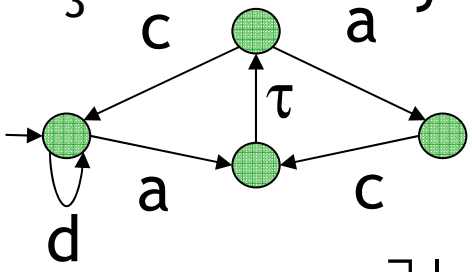
- The algorithm applies on synchronization networks, a general model similar to MEC and FC2 networks
- We implemented the translation into networks for
  - CCS, CSP, LOTOS, mCRL parallel composition
  - E-LOTOS generalized parallel composition and *m* among *n* synchronization
- The translation can still be done for other operators

Limitation 2 solved

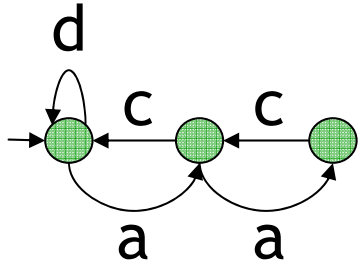
- Interface constraints induced by **any combination of** processes can be computed
- Example: in



the interface I of P_3 induced by P_1 and P_2 is:

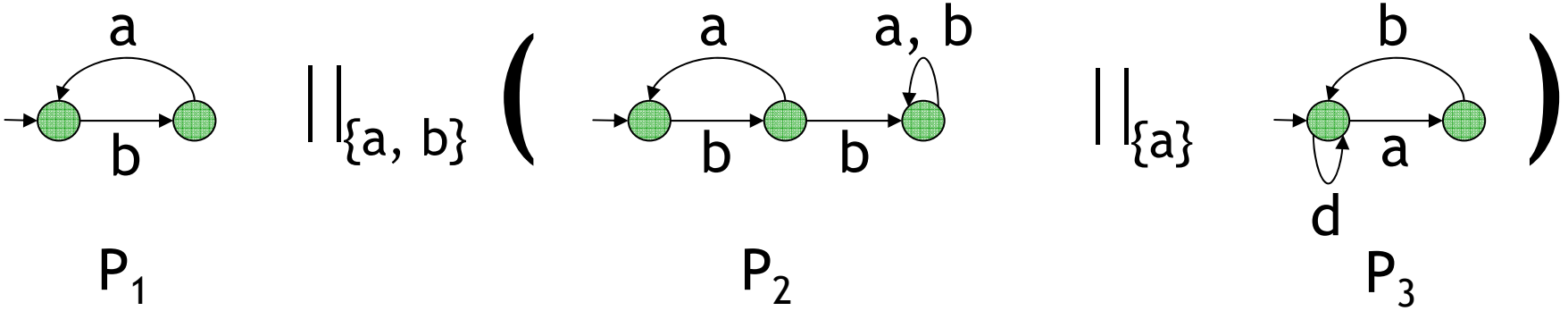


- It yields reduction of P_3 as $P_3 \lceil|_{\{a, c, d\}} I =$

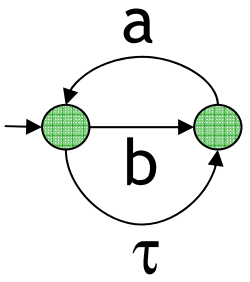


Limitation 3 solved

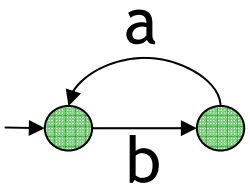
- Interfaces are precise even in presence of nondeterministic synchronization



- The interface I of P_2 induced by P_1 is:



- It yields reduction of P_2 as $P_2 \lceil|_{\{a, b\}} I =$



Implementation in CADP

- Algorithm implemented in Exp.Open 2.0 (-interface option)
- Example: odp.exp

```
hide all but WORK in
        par EXPORT, IMPORT in
                par WORK #2 in
                        "object_1.bcg"
                ||      "object_2.bcg"
                ||      "object_3.bcg"
                ||      "object_4.bcg"
                end par
        ||
                "trader.bcg"
        end par
end hide
```

Implementation in CADP

exp.open -weaktrace -interface "5: 1 2 3" "odp.exp"
generator "trader_interface.bcg"

- Generates an interface graph "trader_interface.bcg" induced by the 1st ("object_1.bcg"), 2nd ("object_2.bcg"), and 3rd ("object_3.bcg") graphs in "odp.exp"
- The interface graph can be used to constrain the 5th graph ("trader.bcg")
- Partial order reduction (persistent set method) preserving observable traces is applied

Applications (1/3)

Philips' HAVi Home Audio-Video leader election

- Modeled in LOTOS by J. Romijn (Eindhoven)
- Largest process (404,477 states) was:
  - Reduced downto 365,923 states (182s, 46Mb) using interface obtained by K&M algorithm
  - Reduced downto 645 states (11s, 8.5Mb) using a refined interface

http://www.inrialpes.fr/vasy/cadp/demos/demo_27.html

Applications (2/3)

ODP (Open Distributed Processing) Trader

- Modeled in E-LOTOS by Garavel & Sighireanu (INRIA)
- Uses *m* among *n* synchronization to model the dynamicity of object exchanges
- Trader reduced from 1 M states without interface downto 256 states using a refined interface

http://www.inrialpes.fr/vasy/cadp/demos/demo_37.html

Applications (3/3)

Cache Coherency Protocol

- Modeled in LOTOS by M. Zendri (Bull)
- 5 agents accessing a remote directory concurrently
- No reduction using interface obtained by K&M algorithm
- Remote directory reduced from 1 M states downto 60 states using refined interface
- Directory generated for a configuration with 7 agents (81 states)

http://www.inrialpes.fr/vasy/cadp/demos/demo_28.html

Refined abstraction in SVL

- Refined interface generation and projection can be done easily within the SVL scripting language
- New "***refined abstraction***" operator calls EXP.OPEN and PROJECTOR automatically
- Example:

```
"cache.bcg" = root leaf strong reduction of
  (
    (AGENT_1 ||| AGENT_2 ||| AGENT_3)
    |[GET_LINE_STATUS, PUT_LINE_STATUS]|
    (refined abstraction AGENT_1, AGENT_2 using DIR_ABSTRACT
          of DIRECTORY)
  );
```

Conclusions

- We provided a new algorithm to synthesize interface constraints automatically
- The algorithm solves the 3 limitations of K&M's algorithm
  - It does not depend on a particular input language
  - It permits to take into account constraints induced by a combination of distant processes
  - It permits a finer analysis of synchronization patterns between processes, thus yielding better reductions
- The method is fully implemented in CADP
- It is easy to use thanks to the SVL scripting language
- Experiments indicate possible reductions by several orders of magnitude