Principles of Distributed Test Synthesis based on True-concurrency Models

Claude JARD

IRISA/CNRS

Campus de Beaulieu

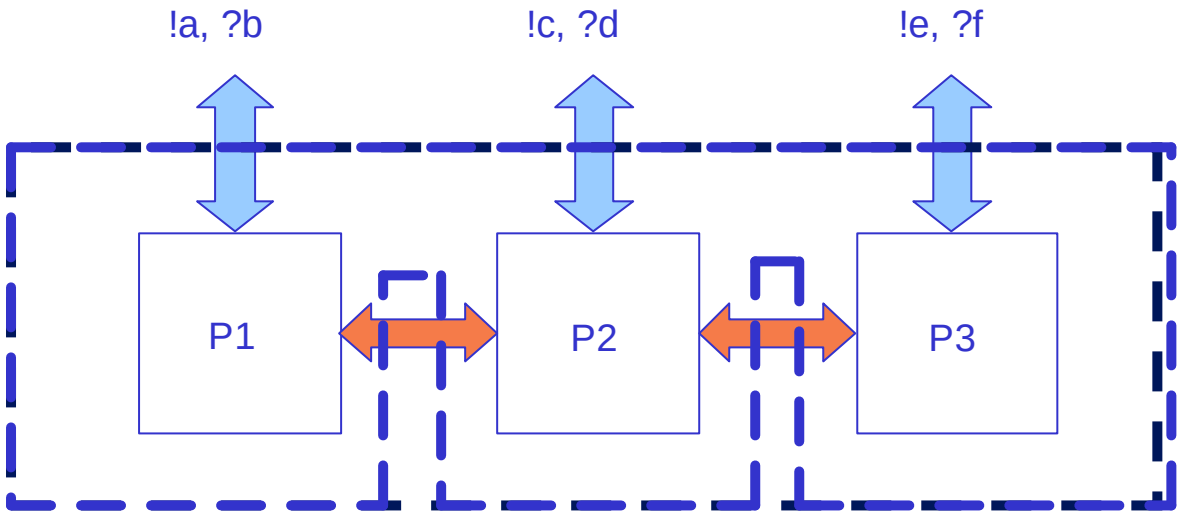
F-35042 Rennes France

Claude.Jard@irisa.fr

http://www.irisa.fr/triskell/perso_pro/home.html

IRISA

Distributed conformance testing

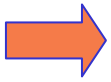


Why synthesis of parallel test cases?

- Required by some test architectures
- Some behaviours need to be tested in parallel (real-time testing, stable states)
- More compact and clear test cases (parallel vs interleaved)
- Existing parallelism in TTCN-3 or MSC notations, not yet fully exploited from the synthesis point of view

Possible approaches

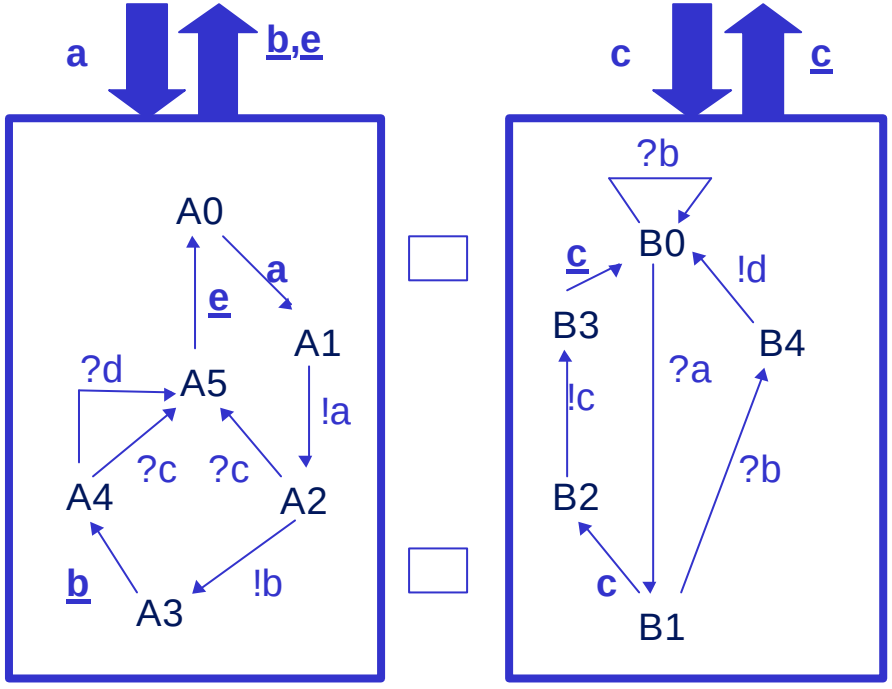
- Distribution of sequential test cases: simple, but does not distinguish between non-determinism and concurrency -> lot of synchronizations
- Keep explicit the parallelism of the specification. Revisit the generation phase by using a true-concurrency model



Context

- This question has been explored for 5 years in IWTCS/Testcom and Forte/PSTV, mainly in Korea (Kim) and Germany (Ulrich, König, Henniger)
- We decided to follow the german approach using unfoldings of nets.
- From our experience, we propose to revisit the complete chain of TGV, using the notion of test purpose, and replacing IOLTS by unfoldings (or event structures).

Small example:
a connect-disconnect protocol



Process A

Process B

Controllable events :

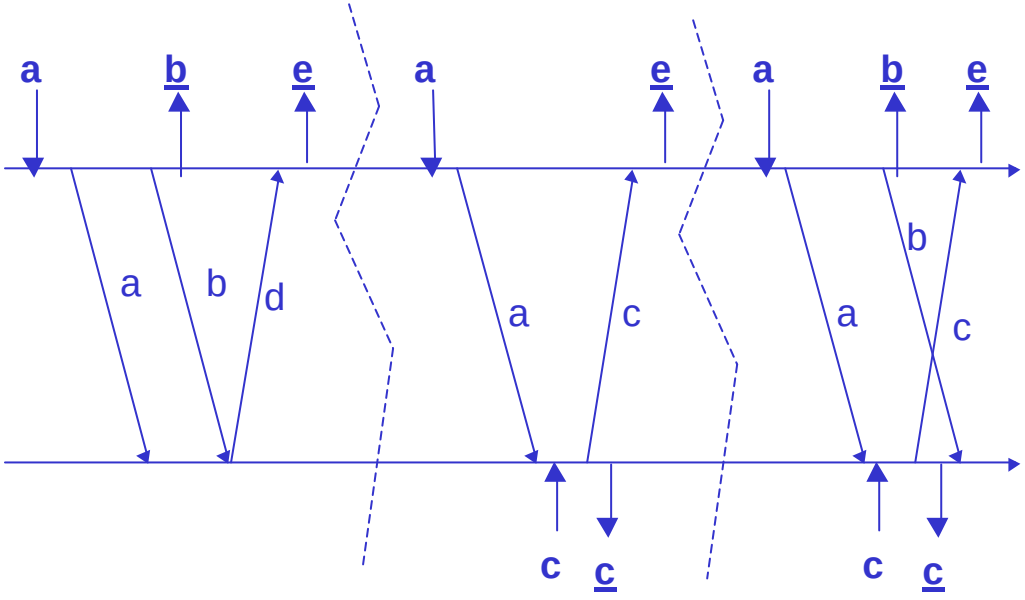
a : Connect_Request
c : Disconnect_Request (from B)

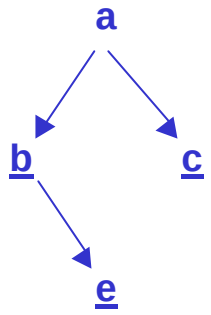
Observable events :

b : Disconnect_Confirm (of A)
c : Disconnect_Confirm (of B)
e : Disconnection_Completed

IRISA

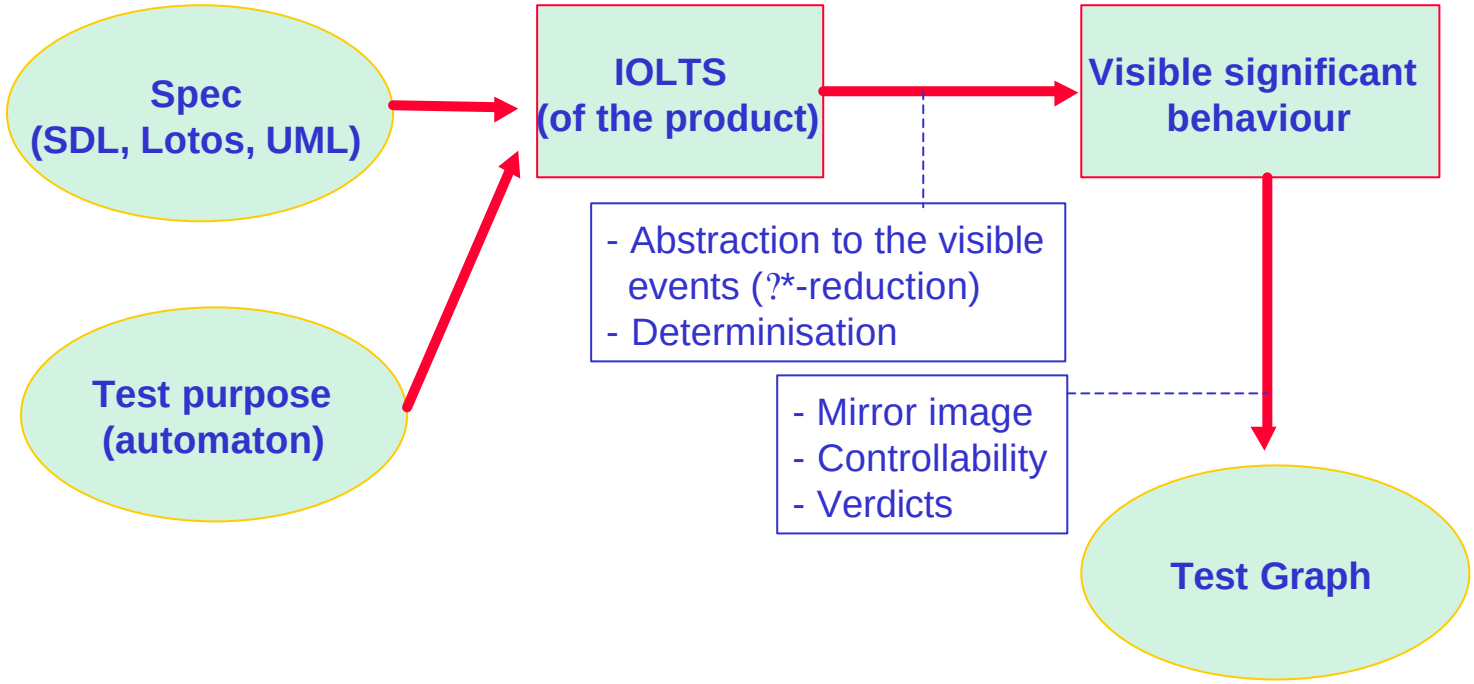
Testing connection-disconnection in case of collision





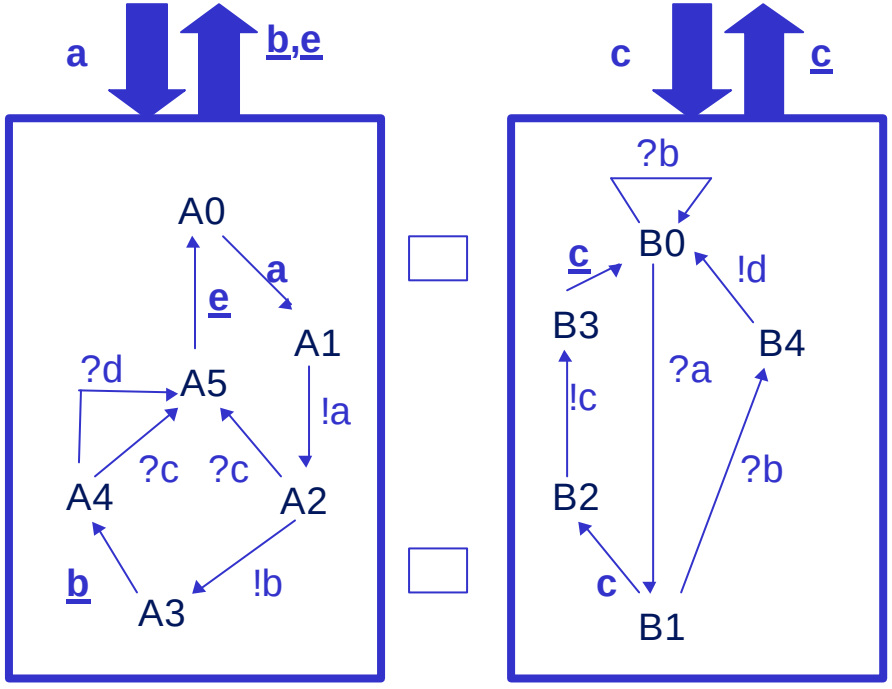
Test purpose :
check the correct disconnection in case of collision
*(**a**,**b**,**c** and **e** must occur, and must be partially ordered as presented)*

TGV at present

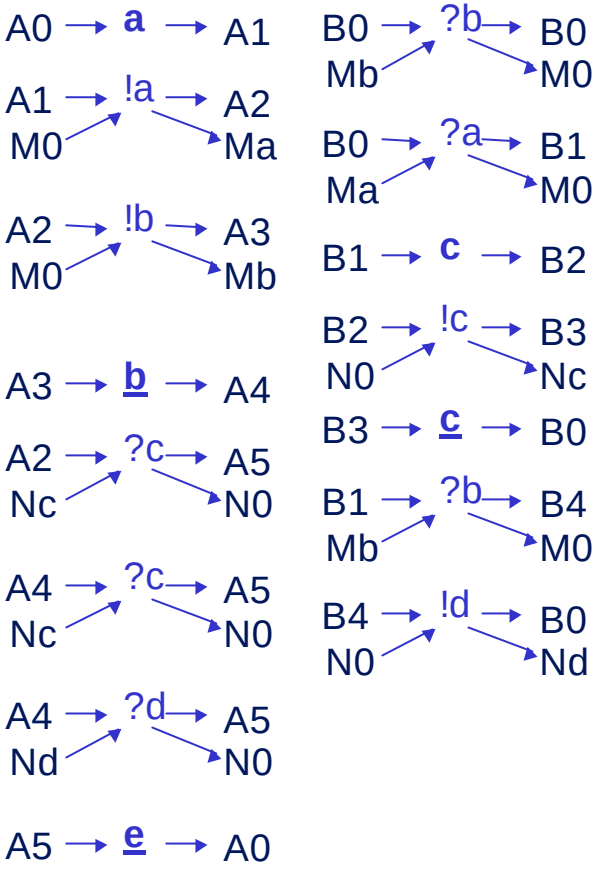


Most of the work is done on-the-fly using APIs

Transitions on partial states: tiles (S)

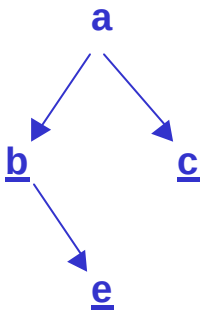


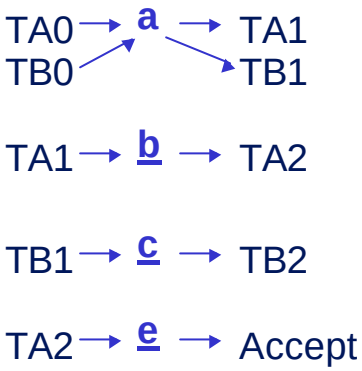
Process A **Process B**



IRISA

Tile system of the test purpose (TP)

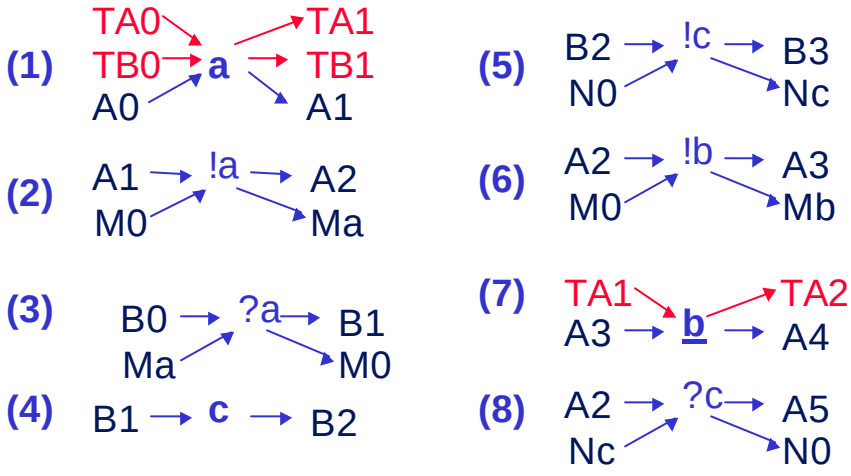




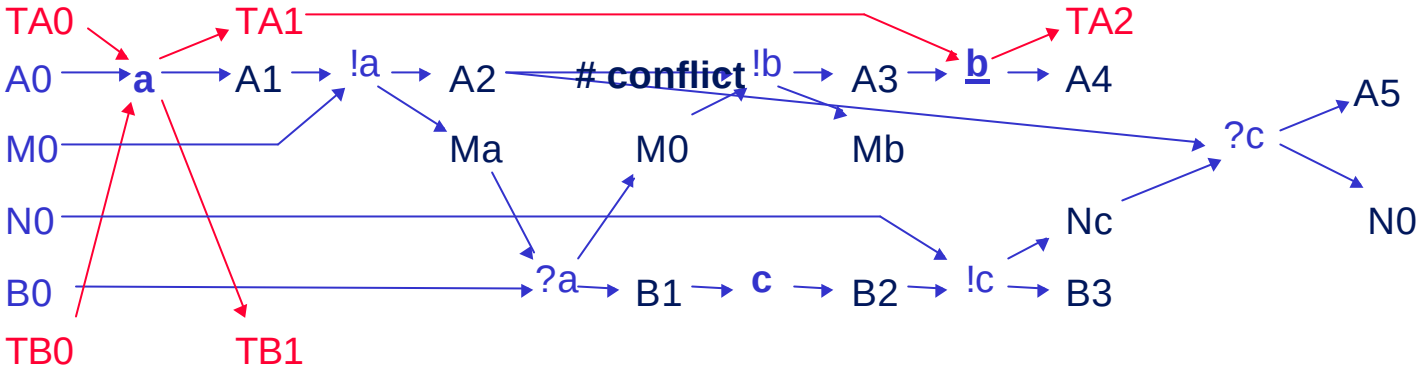
Pre-compilation of the tiles of the product TPxS:

- duplicate the tiles of common actions (making the union of pre- and post-conditions)
- these new tiles are prioritary in case of conflict with the original
- Terminating sink tiles when the post-condition is *accep*t (or *refus*e)

Computing the set of partial order histories: the puzzle game (unfolding)

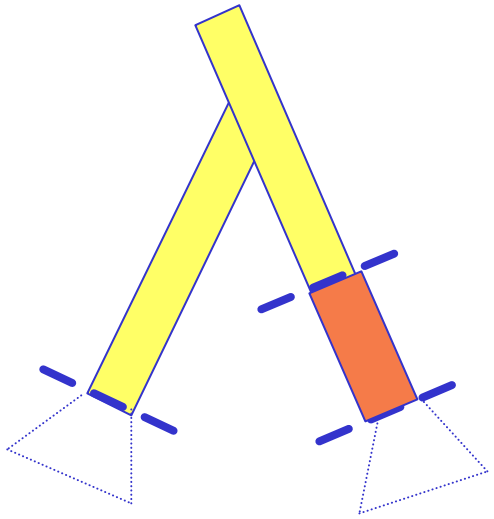


Puzzle := initial_global_state;
repeat
  if it exists a tile t *such that* pre(t) *is a co-set of Puzzle then*
    append t+post(t) *to Puzzle*
forever



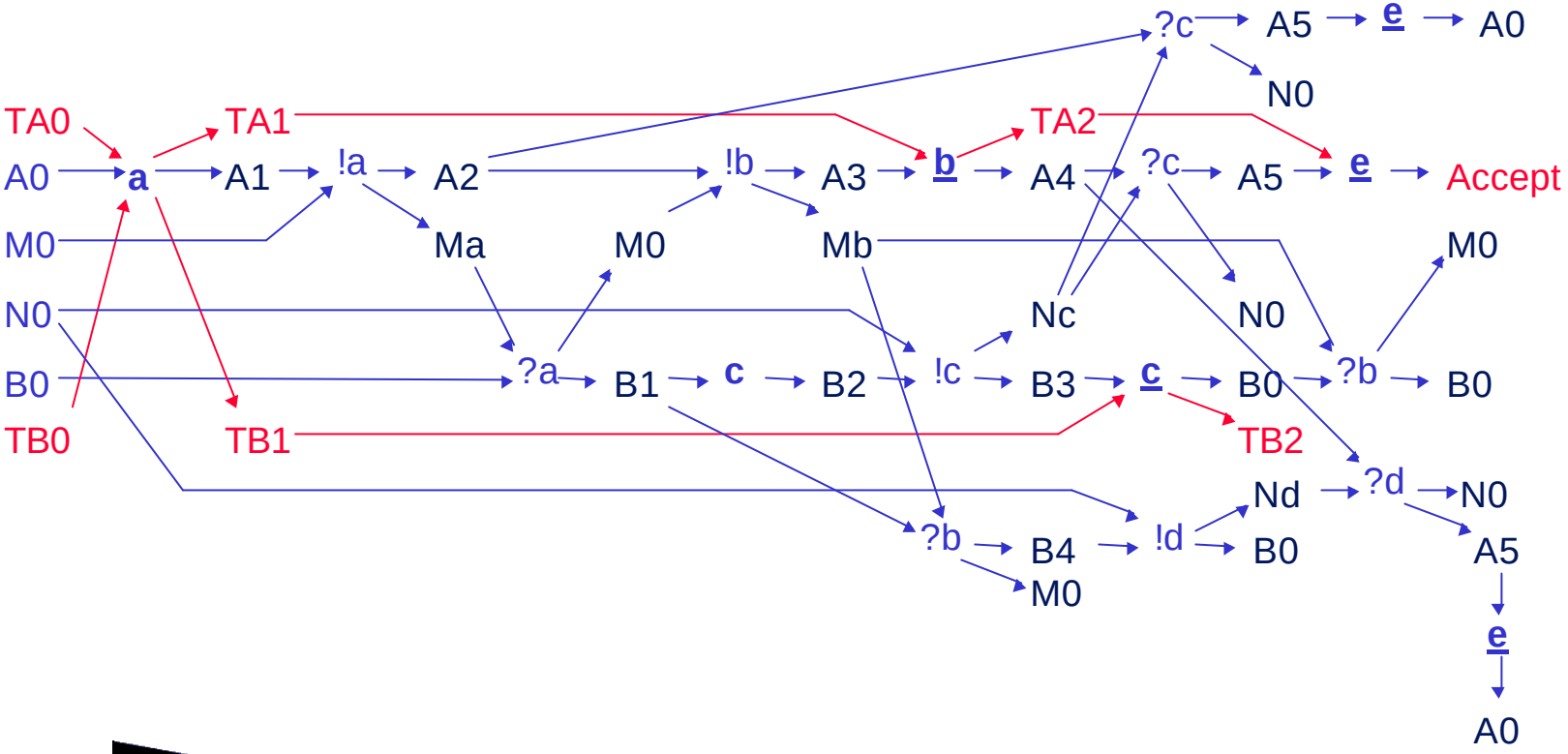
Computing a finite complete prefix
(contains all the possible tiles and configurations)

Finite_Puzzle := initial_global_state;
cut_off := {};
repeat
  Select a tile t *such that* pre(t) *is a*
    co-set of Finite_Puzzle;
  live := t *exists and* pre(t)*? cut_off={};*
  if live then append t+post(t) *to Finite_Puzzle;*
    if ? u < t : Gstate(u)=Gstate(t)
      then cut_off := cut_off ? {post(t)}
until not live



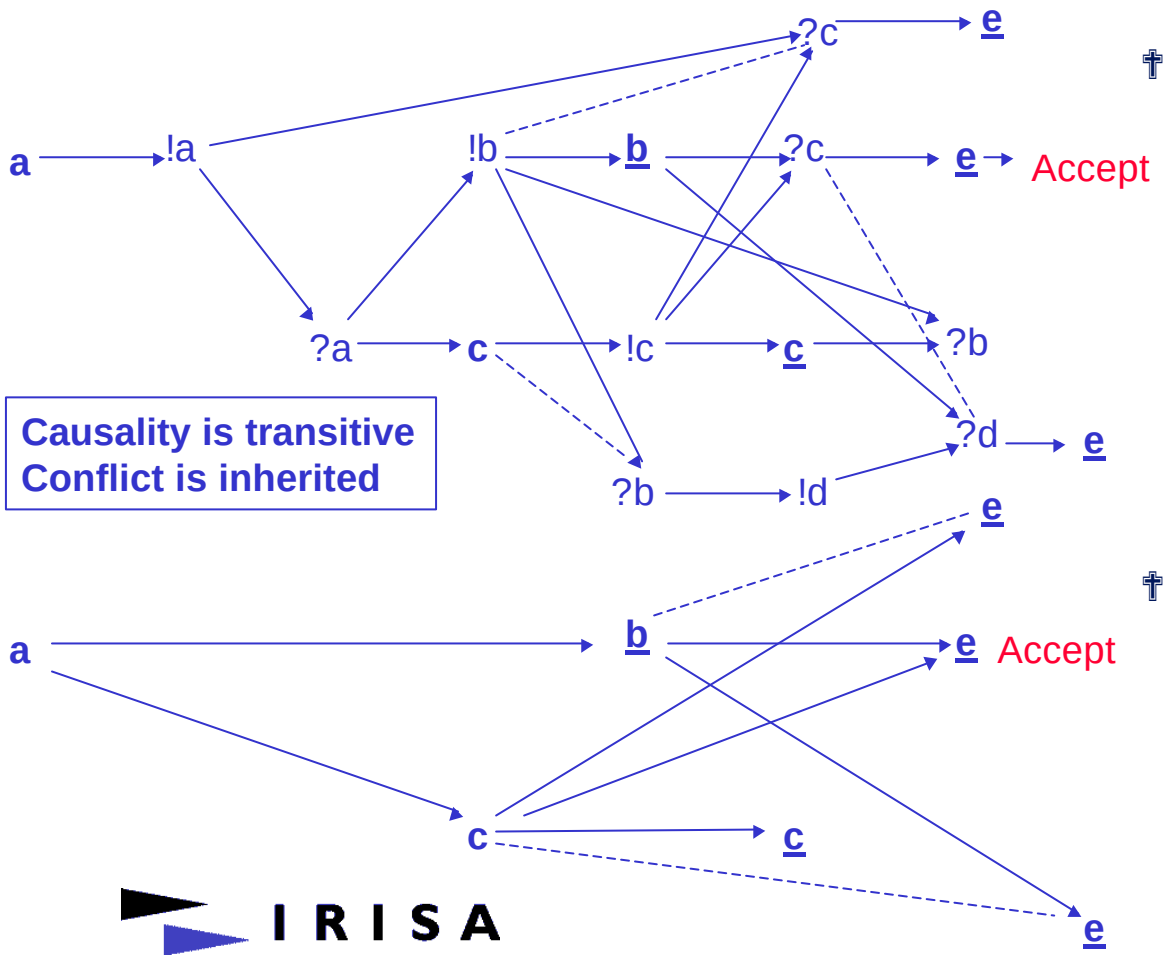
- < can be extended to decrease the size of the structure (cf. Mac-Millan, Esparza), which can be not bigger than the global state representation
- complexity |C|?/? ?(|C| number of conditions, ? degree of //) [notion of canonical prefix]

The finite complete prefix of our example



IRISA

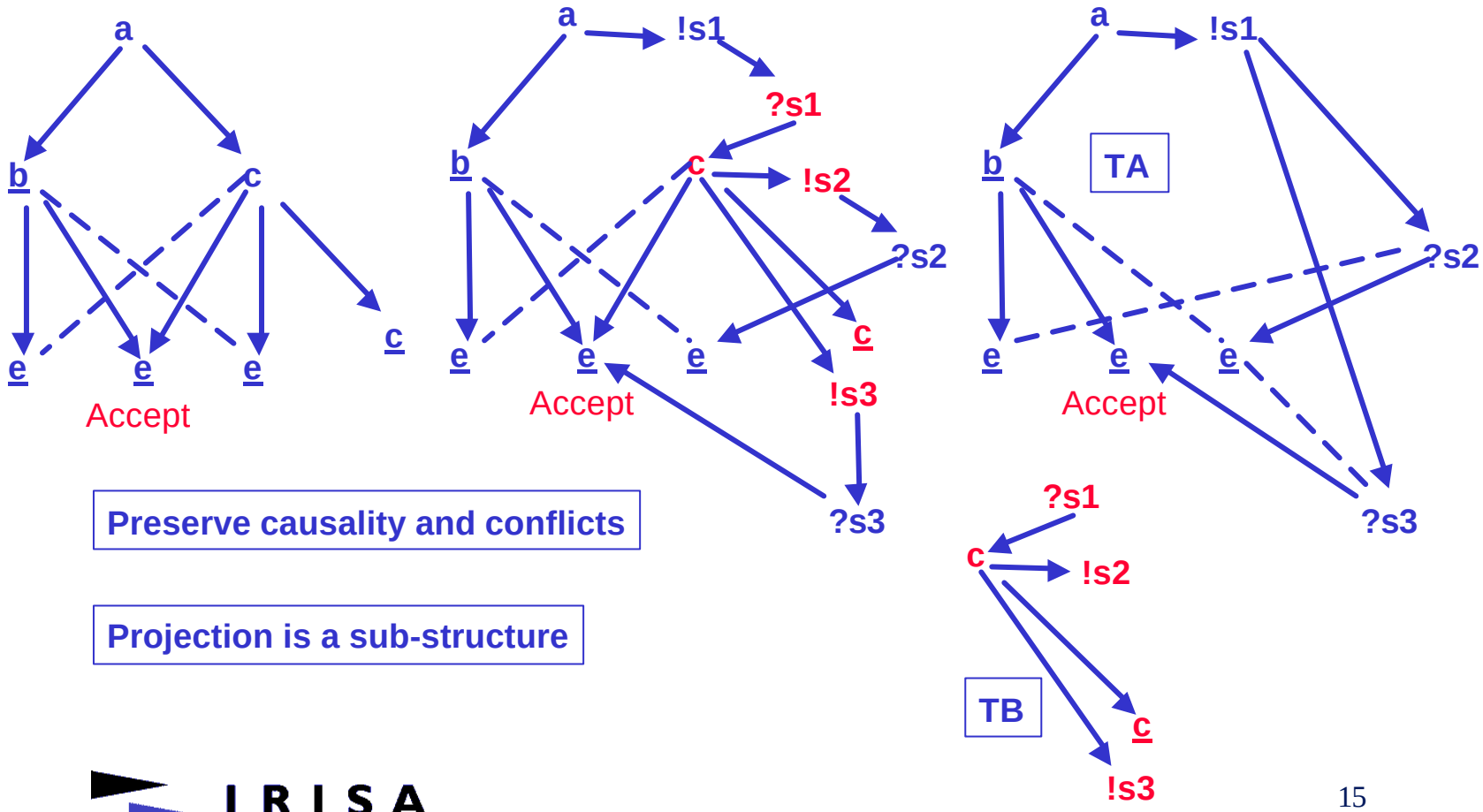
The causal and conflict relations
(covering of the underlying event structure)



- Conditions are just pins for the construction and can be removed
- Abstraction to visible events can be defined as a sub-structure

IRISA

Insertion of synchronisation + distribution by projection

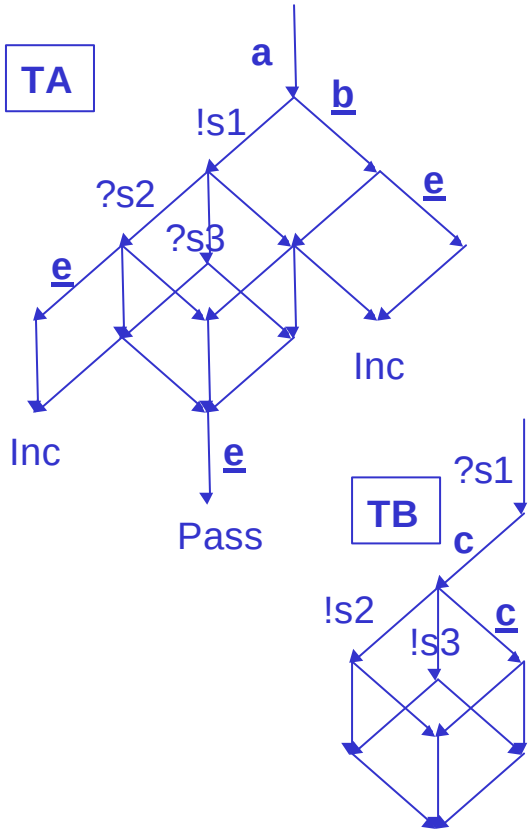


Preserve causality and conflicts

Projection is a sub-structure

IRISA

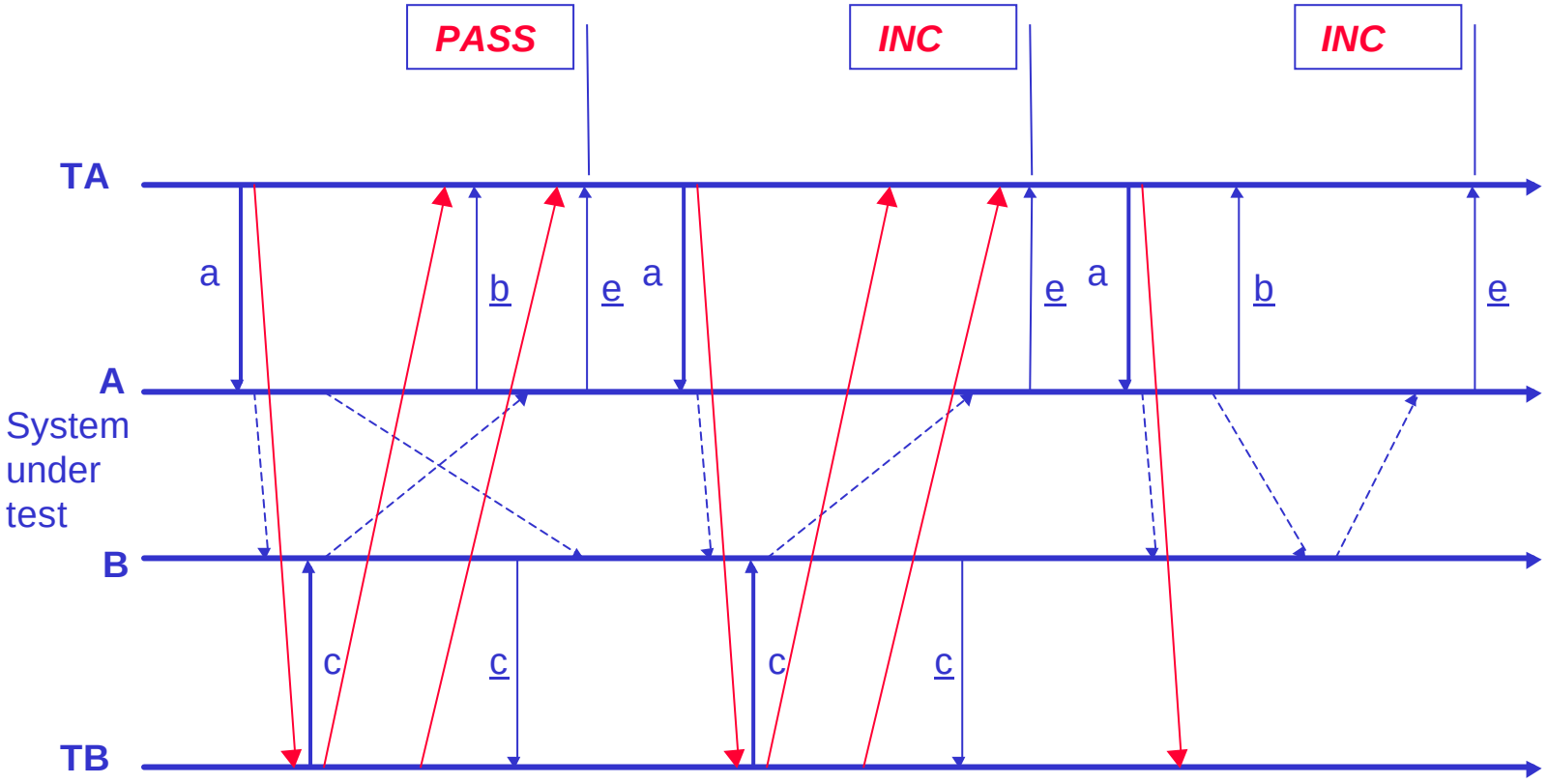
Construction of the test automata



- Build the “budding” lattice of the event structures (the set of configurations: compute the interleavings and take into account the conflicts)
- There exist linear algorithms
- Local controllability of synchronisation messages could be applied (restriction of concurrency)
- Fail upon non-specified receptions

IRISA

Test examples

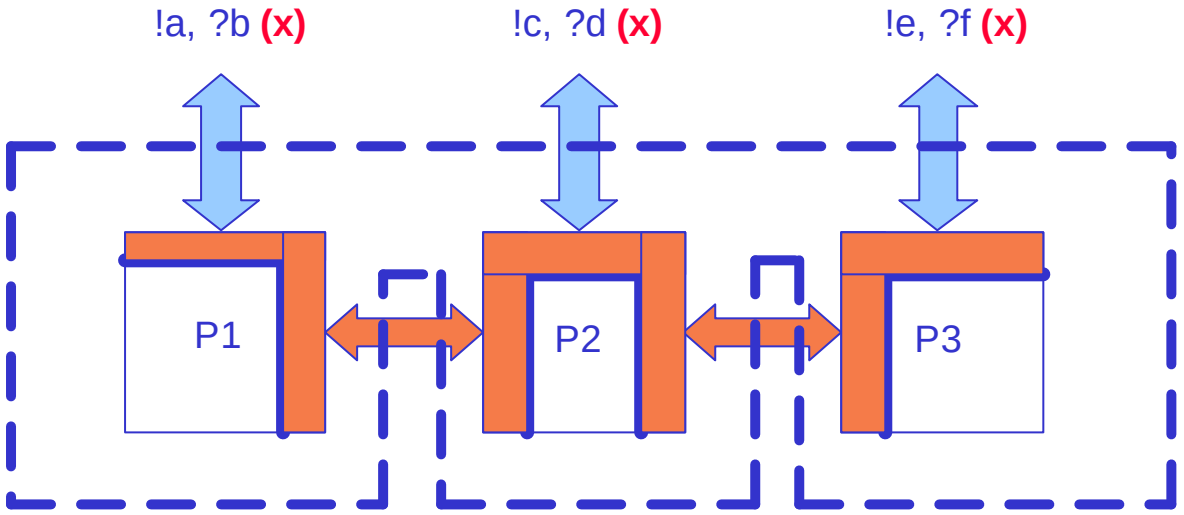


IRISA

Conclusion and perspectives

- Small prototype under development using the MC tool kit of T.U.M. (Römer, Esparza): unfolding of safe Petri-nets (wwwbrauer.informatik.tu-muenchen.de/theorie/KIT)
- Soundness (in the sense of io-conformance is achieved) since all the transformations are trace-preserving
- Definition of a new distributed conformance relation based on partial orders (requires to observe concurrency)
- On-the-fly unfolding and abstraction
- Experiment with the partial order semantics of the action semantics of UML
- Link to symbolic TGV using symbolic tiles

Grey box test architecture



Vector clocks instrumentation:

Local observation : H[i]:=H[i]+1; Timestamp with H
Sending of a message : piggyback with H
Receiving message m(H') : H:=max(H,H')