Using on-the-fly verification techniques for the generation of test suites

J.-C. Fernandez (Spectre Verimag Grenoble France),

C. Jard (Pampa Irisa Rennes France),

T. Jéron (Pampa Irisa Rennes France),

G. Viho (Pampa Irisa Rennes France)

http://www.imag.fr/VERIMAG/
http://www.irisa.fr/pampa/

Conformance testing

- **Domain:** reactive programmes (e.g. protocols)
- **Problem:** check that an implementation conforms to its specification.
- **Black box testing:** the implementation is only visible by its interactions with its environment.
- **In practice:** stimulate the implementation, observe its behaviour, and try to deduce its conformance or not to the specification. => error detection.

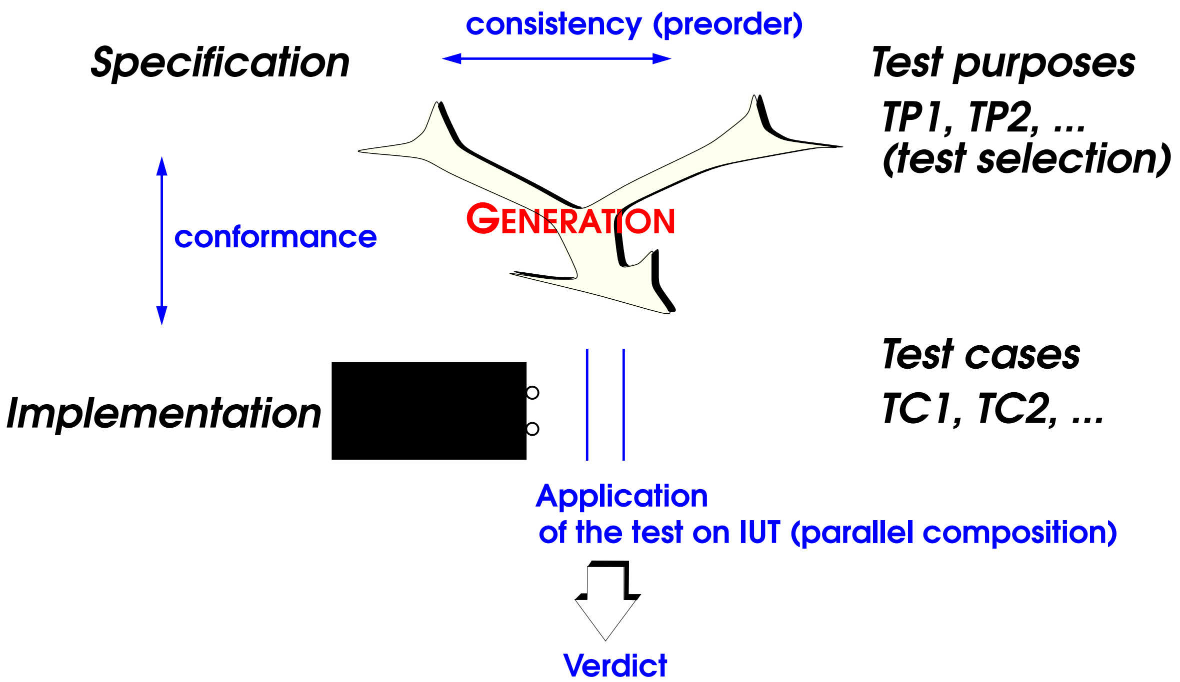
Motivations

- **Industrial context:**
  - most conformance tests are written by hand ⇒ high cost.
  - no clear definition of conformance ⇒ biased verdicts.

  Thus automation from formal specifications can be profit earning
- But academic research does not help enough:
  - unrealistic hypothesis
  - algorithmic complexity / size of real specifications
  - no significant case study
  - do not consider the industrial expertise

- USE FORMAL SPECIFICATIONS (LDS, LOTOS, ESTELLE, ...)
- MODELIZE THE INDUSTRIAL EXPERTISE
- TRY TO AVOID STATE EXPLOSION BY USING VERIFICATION TECHNIQUES

Main principles



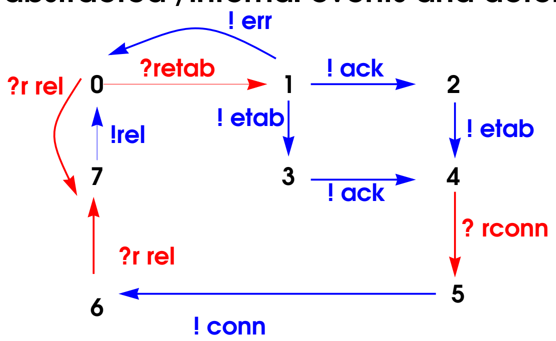
Modelization by input/output labelled transition systems

- Specification S
- Implementation I
- Test purpose TP
- Test case TC

modelized by input/output labelled transition systems (IOLTS)

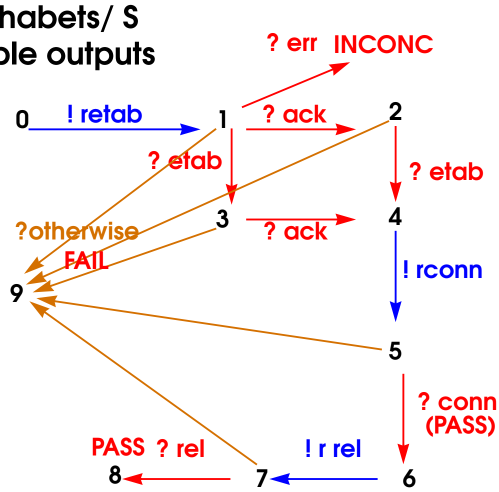
An IOLTS is a 5-uple $(Q,A_I,A_O,!\ \ ,q_{init})$ with
- Q a finite set of states, q_{init} an initial state
- A_I and A_O are the input and output alphabets
- $A = \{?\} \times A_I \cup \{!\} \times A_O$
- $! \ \ \subseteq Q \times A \times Q$ is the transition relation.

S: IOLTS of the specification abstracted /internal events and determinized

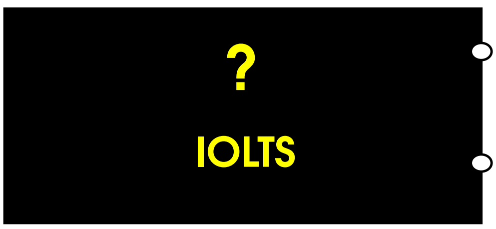


TP: deterministic and acyclic IOLTS with controlable inputs

TC: acyclic and deterministic IOLTS
miror alphabets/ S
controlable outputs
verdicts
timers



I: IOLTS



Consistency of TP / S

- A **Test Purpose** is a property of some sequences of interactions of S that one particularly wants to test.
  The **purpose** is the **selection** of some particular behaviour of S.
  Thus it is not suposed to be satisfied by all sequences.

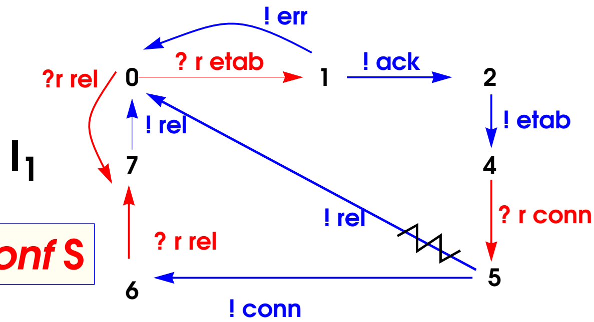
- **Consistency:** weak notion of satisfaction (preorder):

  TP is consistent with S if the set of behaviours of TP is included in the set of behaviours of S and from each state of S corresponding to an Accept state of TP, there is a path to the initial state of S.

Conformance

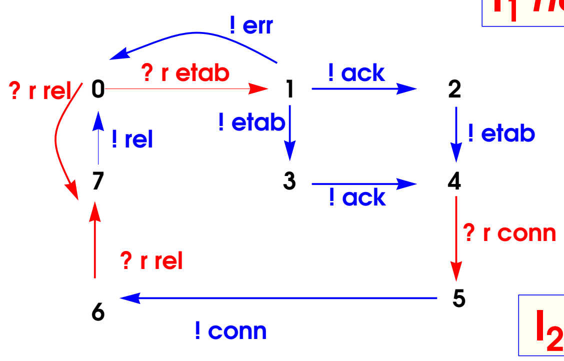
I ioconf S
iff
$\forall \sigma \in$ **traces(S),**
$O($**I after** $\sigma) \subseteq O($**S after** $\sigma)$

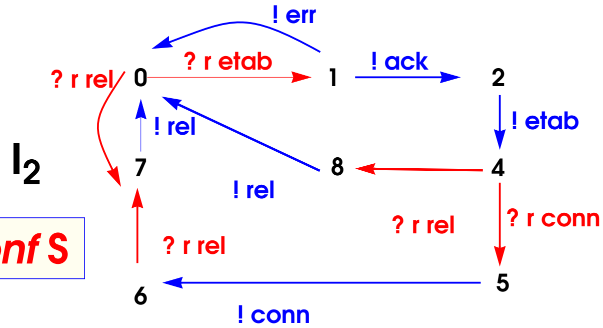
Implementations



I_1 *not ioconf* S

Specification S





I_2 *ioconf* S

Principles of the algorithm

- Algorithm described by operational rules (à la Plotkin).

- Construct a DAG by a DFS of a product between S and TP:

  push :

  - consistency checking
    by testing the reachability of states in (Accept, $q^S{}_{init}$)).

  pop :

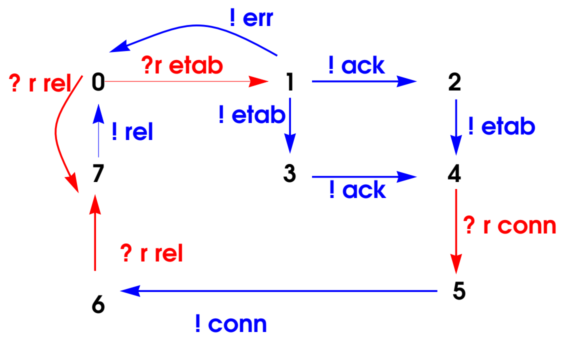
  - synthesis of a DAG from states in (Accept, $q^S{}_{init}$)
    while assuming controlability,
    decorating some transitions with verdicts
    computing the miror image

- Second DFS of the resulting DAG
  for the placement of timers operations (Start, Cancel, Timeout).

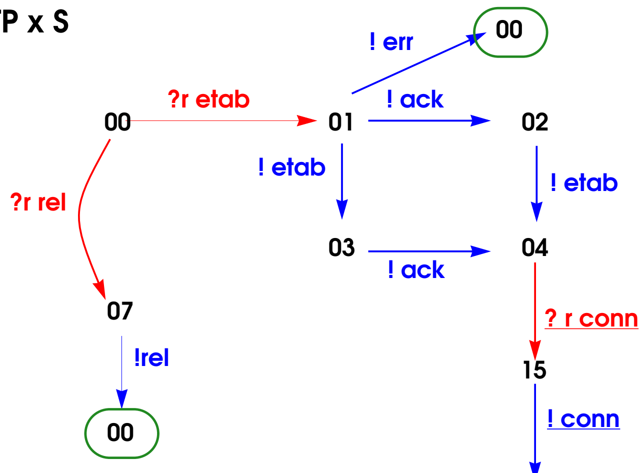
Product : test purpose x specification

Test Purpose TP

Specification S



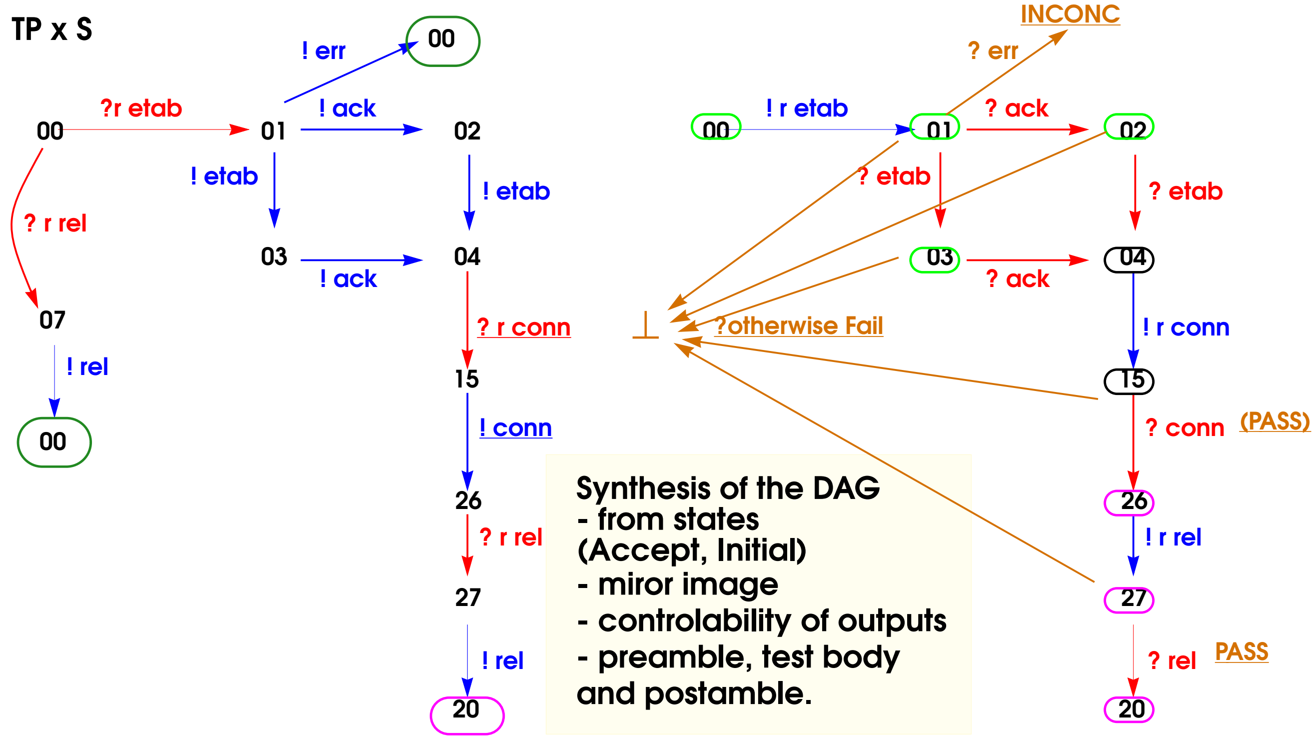
TP x S



DAG: acyclic IOLTS

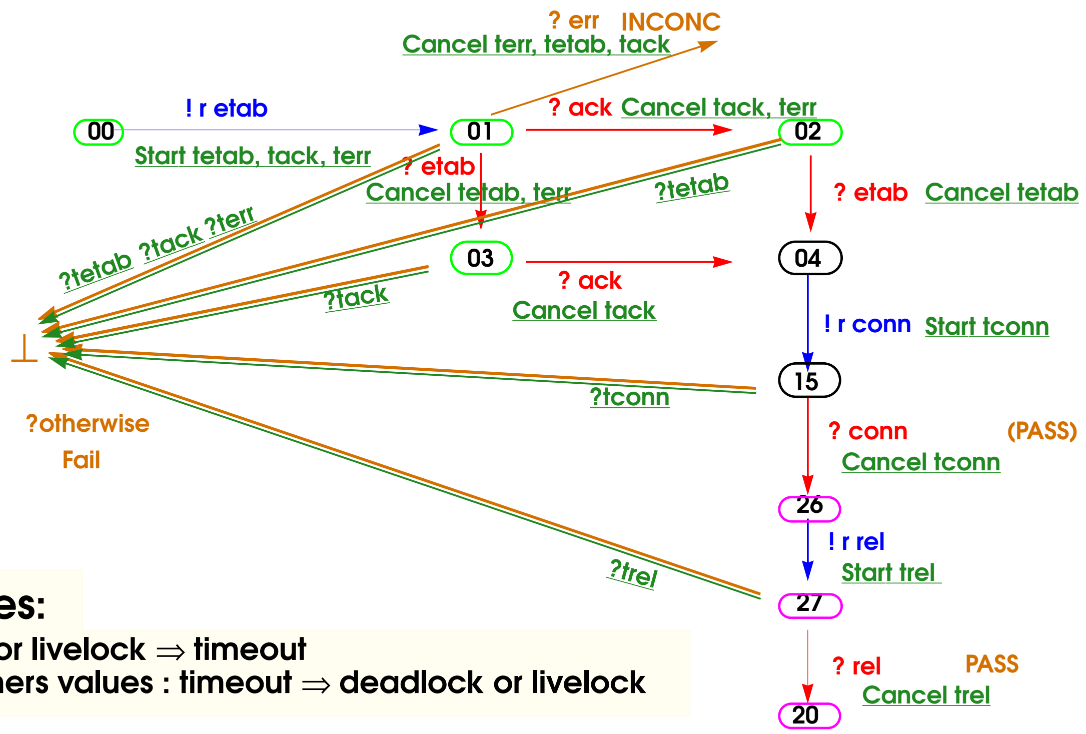
- cut loops
- stop in states (Accept,$q^S{}_{init}$)

Synthesis of a DAG by DFS of TP x S



Synthesis of the DAG
- from states (Accept, Initial)
- miror image
- controlability of outputs
- preamble, test body and postamble.

Management of timers



Properties:

deadlock or livelock $\Rightarrow$ timeout

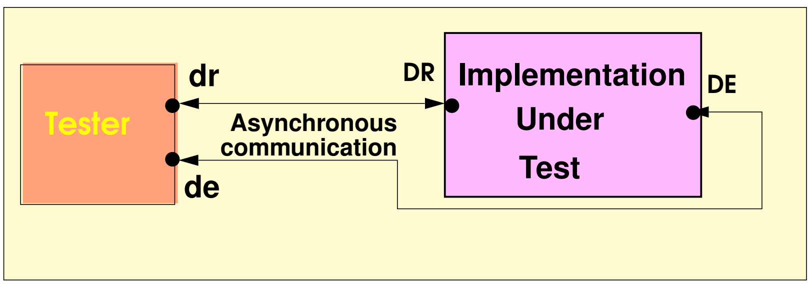
hyp. on timers values : timeout $\Rightarrow$ deadlock or livelock

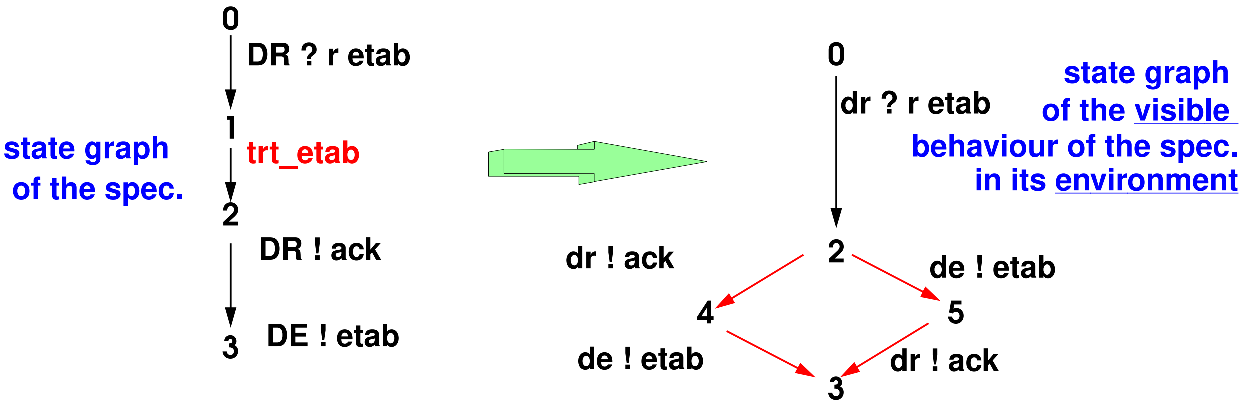
Theoretical results

- **Complexity** : linear in time and space / size(**TPxS**)
- **Consistency checking:** the algorithm applied to **TP** and **S** returns true and constructs a test case **TC(TP,S)** iff **TP** is consistent with **S**.
- **Unbiased test cases: hypothesis on timers values**
  TC(TP,S) applied to **I** produces a **FAIL** verdict ⇒ **not(I ioconf S)**.
- **"Exhaustivity"** :
  BF(I): bounded fairness hypothesis on **I** :
  $\forall K, \exists \mathbf{n}, \forall \sigma \in (\{?\} \times A_O^S \cup \{!\} \times A_I^S)^*, |\sigma| \leq K,$
  $(\sigma \parallel I)^n$ shows all its possible behaviours.

  ∀ **I, BF(I) and I not ioconf S** ⇒
  ∃ **TP** consistent with **S** such that
  the algorithm **may** produce a test case **TC(TP,S)**
  which produces a **FAIL** verdict if applied **n** times to **I** .

Asynchronous testing





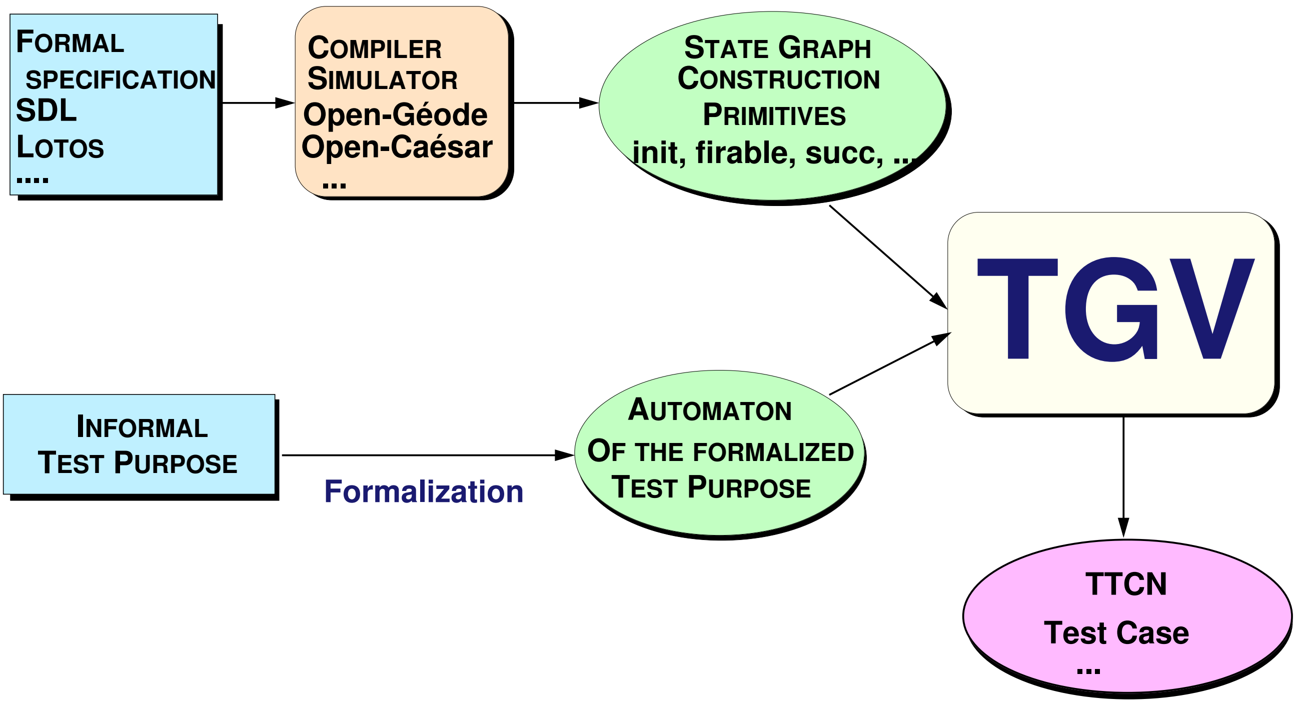
On-the-fly generation

- **Problem:** state space explosion when producing the IOLTS of the specification from an FDT.
- **One solution:** constructing test cases during the construction of the IOLTS of S.
- **How ?**

) access to primitive functions of the graph construction of a simulator produced by the compiler (init, firable, succ).

) integration of abstraction of internal events and determinization during the graph construction.

TGV (IRISA- Verimag)



Experimentations

- ❏ Project for the French Army (Verilog, Cap-Sesa, Cnet, Inria)
  - DREX protocol, a military ISDN protocol.
  - data: SDL spec. + test purposes + manual test cases
  - generating tests and comparing with manual tests.
  - Results : identification of some errors in manual test cases, efficiency of the generation
- ❏ Other case studies in progress:
  - Fast reservation protocol of the ATM in SDL.
  - Memory cach protocol in Lotos (Bull).

Perspectives:

- ❏ Design of a commercial tool based on ObjectGEODE (Verilog), integrating some advantages of TGV
  on-the-fly generation, consideration of the test environment, management of timers.
- ❏ Expressiveness of Test Purposes : which temporal logic ?
- ❏ Limit the state explosion: partial order methods combined with on-the-fly generation ?
- ❏ Interoperability testing: distribution of a tester into parallel testers.