Testing Behavioural Properties of Systems using Model-based Verification Techniques

Claude JARD
IRISA/CNRS Rennes France

- Outline
  - Introduction to conformance testing (protocol area)
  - Brief survey of popular methods (FSM based, LTS based)
  - Extending on-the-fly model-based verification techniques

Conformance testing

- **Domain:** reactive programs (e.g. protocols)
- **Problem:** check that an implementation conforms to its specification (single entity, service, interoperability, ...)
- **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

Who tests conformance ?

- ❑ Developers, providers, before market
- ❑ Users to check the correct behaviour
- ❑ Telecom suppliers before connecting a product to the network to prevent crashes
- ❑ Independent test laboratories accredited to provide implementation certification

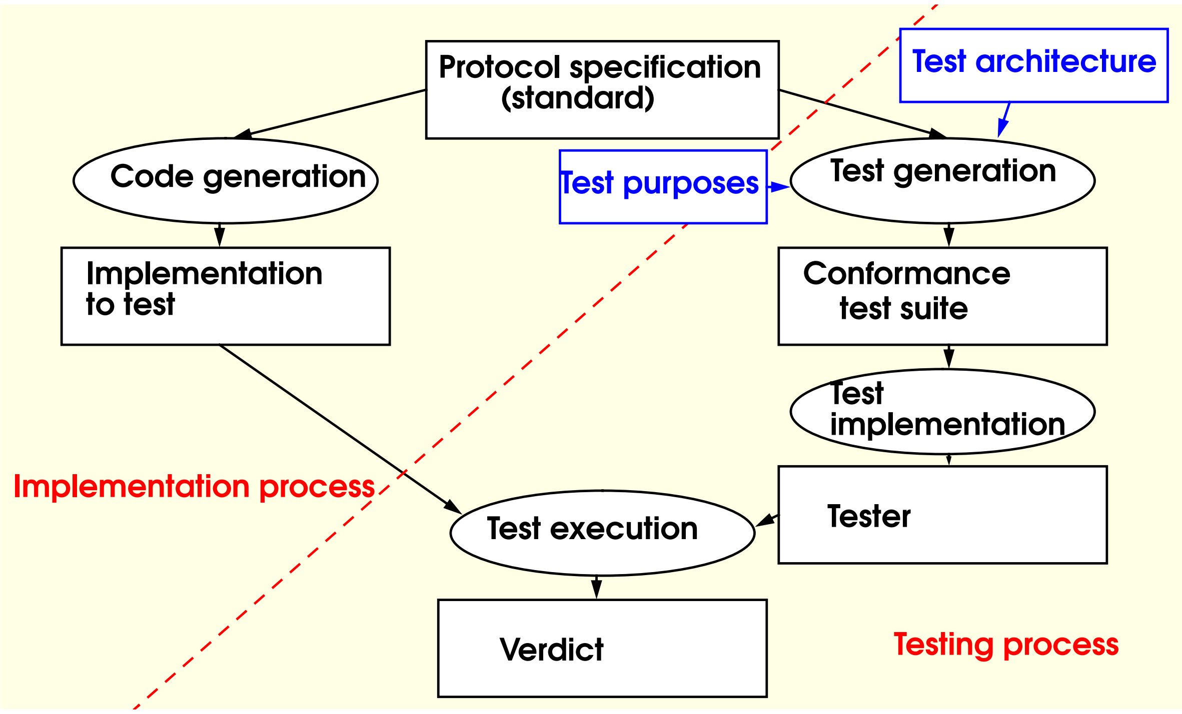
---

The ISO 9646 methodology

❑ OSI Conformance Testing Methodology and Framework. Originally developed for testing ISO protocols, it applies for a large variety of systems. It is composed of several parts :

- Introduction and general concepts
- The test specification process
- Definition of TTCN notation
- Execution of tests
- Requirements of test laboratories and clients during the conformance checking

IS-9646 methodology



Protocol specification (standard)

Code generation

Implementation to test

Test purposes

Test architecture

Test generation

Conformance test suite

Test implementation

Tester

Test execution

Verdict

Implementation process

Testing process

Local test method

Distributed architecture

Claude JARD

Nr	Lab.	Behaviour description	Refs.	Verdict

Motivations for formal conformance testing

❑ **State :**

- most conformance tests are written by hand ⇒ high cost (a complete test suite is composed of several hundred test cases. Writing test cases is long and repetitive)
- no clear definition of conformance ⇒ biased verdicts
- 20% of test cases are considered erroneous

Thus automation from formal specifications can be profit earning

❑

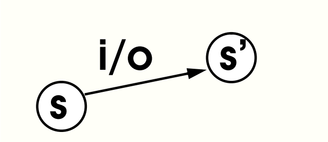
- USE FORMAL SPECIFICATIONS (SDL, LOTOS, ESTELLE, ...)
- FORMALIZE THE TEST PROCESS
- DESIGN ALGORITHMS FOR AUTOMATIC GENERATION

Brief survey on the algorithmic aspects of automatic generation based on FSM

- ❑ Using automata, languages and traces (circuits)
- ❑ Algebraic methods : preorders and behavioural equivalences (“test theory”)

Automata based methods

- ❑ Specifications and implementations are modelized by Mealy machines (input-output FSM)



- ❑ **Principle :** test each transition of the specification and linked all these tests into a single test sequence
- ❑ **Single transition testing :**
  - bring the automaton in state s,
  - apply input i and check the output o (detects output errors)
  - identify s’ (detects transfert errors)

Variants (most popular)

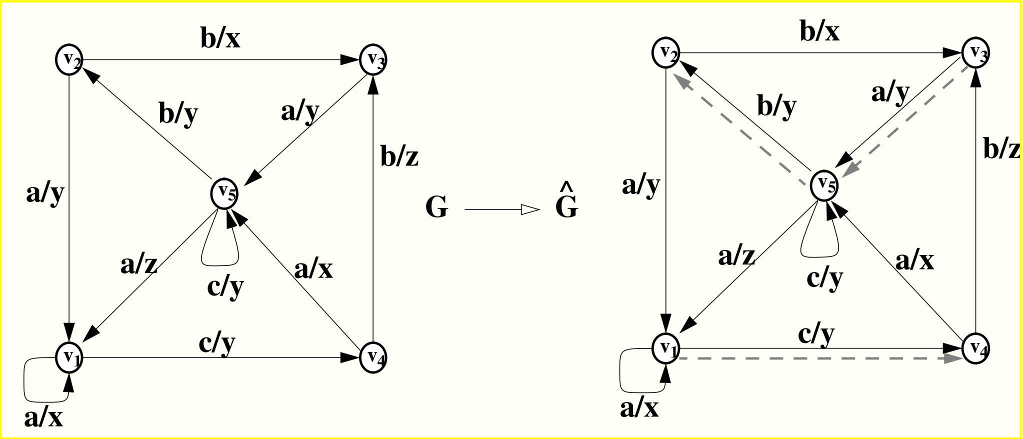
- **Transition tour :**
  - no state identification
  - strong connexity is required
  - Chinese postman problem : find an Eulerian path of a symmetric augmentation of the automaton. Polynomial.
- **Unique Input Output sequences (UIO) :**
  - a sequence UIO is associated with each state s’ which distinguishes s’ among the other states
  - minimality and strong connexity is required
  - Rural Chinese postman problem. Polynomial.

Others

- **Distinguishing sequences (DS) :**
  - state identification
  - does not exist in general
- **Characterizing sequences (W) :**
  - state identification s’ by a set of sequences W such that for all s”, it exists cs in W that distinguishes s’ from s”
  - for minimal and completely specified machines

Transition tour

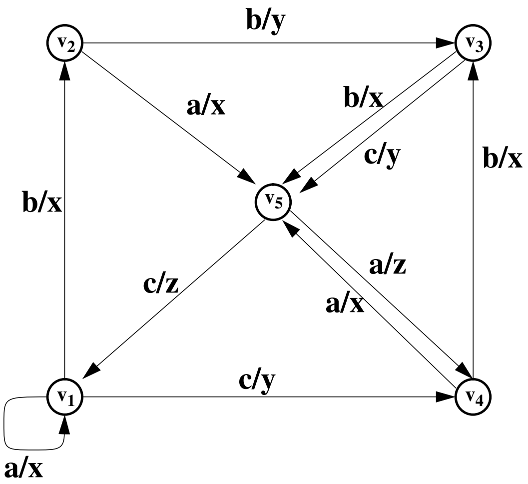
- Chinese postman : find a circuit of minimal length containing all the transitions = Eulerian path of a symmetric augmentation
  - symmetric aug. : optimization problem. Polynomial
  - Eulerian circuit : linear a/x.c/y.b/z.a/y.c/y.a/z.c/y.a/x. b/y.b/x.a/y.b/y.a/y



Unique Input Output sequences

Machine de Mealy $G^*=(V^*, E^*)$

UIO : for all vj, w*(vi,UIO(vi)) <> w*(vj,UIO(vi)))



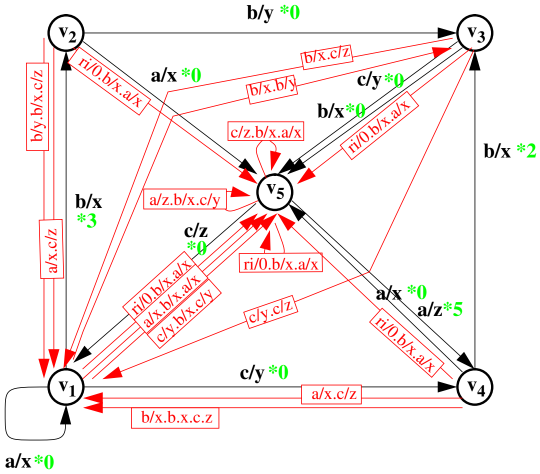
UIO often exist
Exponential complexity

v_1: b/x.a/x
v_2: b/y
v_3: b/x.c/z
v_4: b/x.c/y
v_5: c/z

$\text{Test}(v_1,v_2,b/x) = b/x.\ \text{UIO}(v_2) = b/x.b/y$
$\text{Test}(v_2,v_3,b/y) = b/y.\text{UIO}(v_3) = b/y.b/x.c/z$
etc...

PB : find a sequence of minimal length containing every elementary tests

UIO (continued)



$G' = (V', E')$ avec $V' = V$ et $E' = E \cup E_C$

$E_C = \{(v_i, v_k, L_l.UIO_j) \mid (v_i, v_j, L_l) \in E$
et $tail(UIO_j) = v_k\}$

$G^{\wedge *} = (V^{\wedge *}, E^{\wedge *})$: **Rural symmetric augmentation of G'**

Computation of G^* : polynomial

n = |V|, m = |E|, m0 = |Ec|

O(n(m+nlog n).log m0/m)

Optimal test sequence :

ri/0
a/z.b/x.c/y
a/x.c/z
b/x.c/z
a/x.c/z
b/x
ri/0.b/x.a/x
b/x.b/y
c/z.b/x.c/y
c/y.b/x.c/y
b/x
b/x
ri/0.b/x.a/x
a/z
c/y.c/z
ri/0.b/x.a/x
a/z
b/y.b/x.c/z
ri/0.b/x.a/x
a/z
ri/0.b/x.a/x
a/x.b/x.a/x
a/z
b/x
b/x
a/z
b/x.b/x.c/z
c/z.

Advantages and drawbacks

- Automatic generation. Automatic selection of test cases.
- General classes of errors are defined (transfer, output, ...)
- Possibility to define test coverage

BUT

- Unrealistic assumptions on the FSM view of the implementation (number of states for example)
- Requirements on the FSM view of the specification
- Algorithmic complexity => applicable only on small specs (several hundred states)
- Does not fit with the industrial usage (IS-9646)

Algebraic approach

- Issued from process algebras (Hennessy-Brinskma's test theory)
- Specs, implementations and tests are modelled with Labeled Transition Systems. Synchronous communication
- Based on a formal definition of a conformance relation between implementation and specification
- Automatic derivation of a “canonical” tester from the spec.

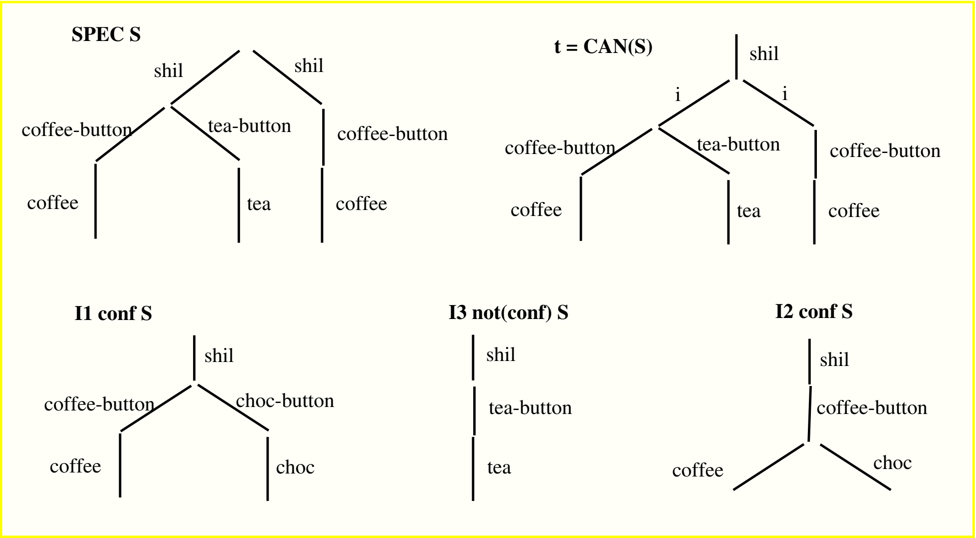
Problems :

- how to select a reasonnable number of test cases?
- originally, does not distinguish between outputs (controlable) and inputs (observables).

=> see however Tretmans' works

Brinskma's canonical tester

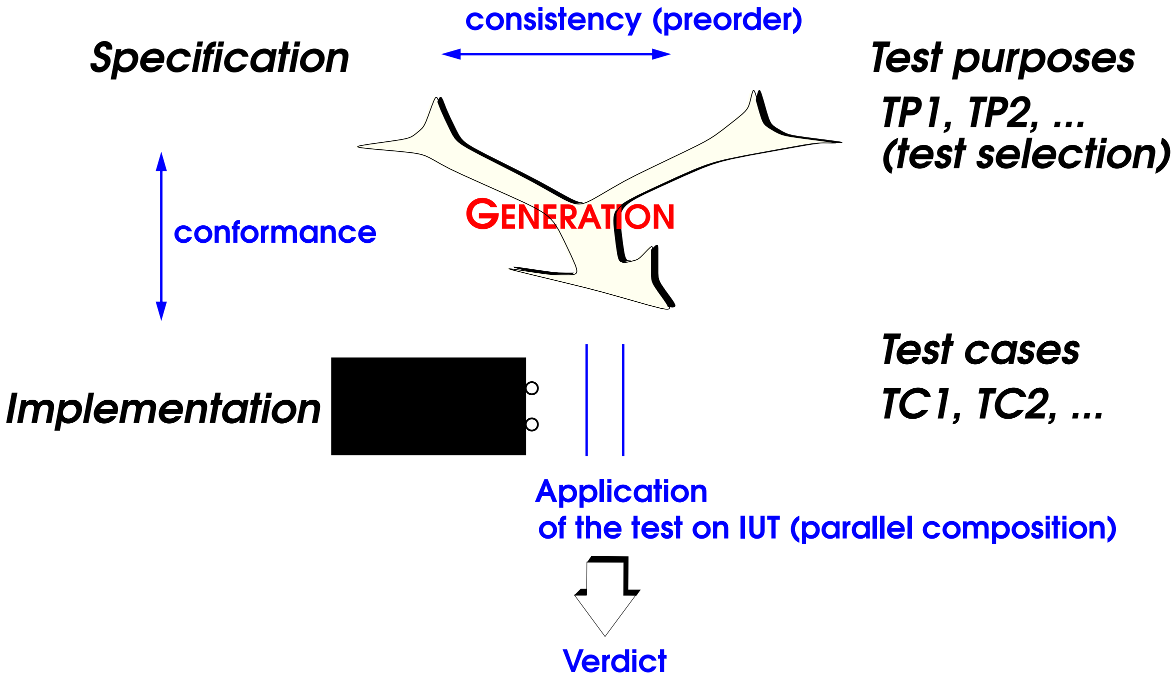
- Spec S, Implementation I, Test t are LTS
- Conformance relation :
  - I conf S iff : for all w in traces(S), for all A subset of L I after w refuses A => S after w refuses A
- Examples :



Using on-the-fly model-based verification techniques : the TGV experiment (IRISA/Verimag)

- ❑ Principles
  - Test objects are modelled by IOLTS (strong distinction between observable and controlable actions)
  - Formal definition of a conformance relation
  - Test selection using test purposes
  - Correction of test cases : (FAIL reports non conformance, non conforming impl. might be rejected by a test)
  - Efficient generation : linear algorithms, on-the-fly generation
- ❑ Industrial experiments and prospects

Main principles



consistency (preorder)

Specification

Test purposes

TP1, TP2, ...
(test selection)

conformance

GENERATION

Test cases

TC1, TC2, ...

Implementation

Application
of the test on IUT (parallel composition)

Verdict

Modelization by input/output labelled transition systems

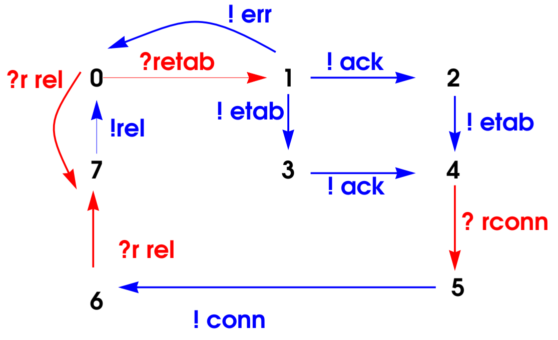
- ☞ Specification S
- ☞ Implementation I
- ☞ Test purpose TP
- ☞ Test case TC

modelized by input/output labelled transition systems (IOLTS)

An **IOLTS** is a 5-uple $(Q,A_I,A_O,! \ , q_{init})$ with

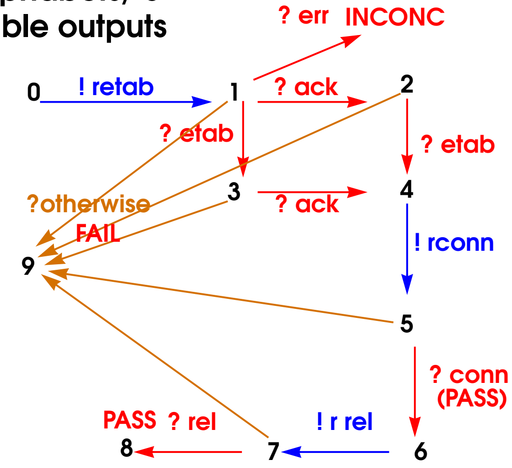
- Q a finite set of states, q_{init} an initial state
- A_I and A_O are the input and output alphabets
- $A = \{?\} \times A_I \cup \{!\} \times A_O$
- $! \ \subseteq Q \times A \times Q$ is the transition relation.

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

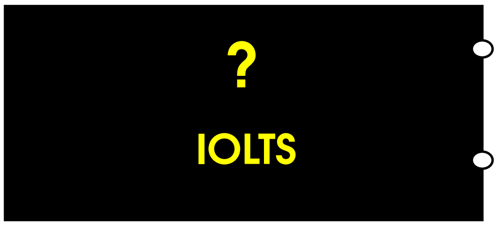


TP: deterministic and acyclic IOLTS with controlable inputs

TC: acyclic and deterministic IOLTS
mirror 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 supposed to be satisfied by all sequences.

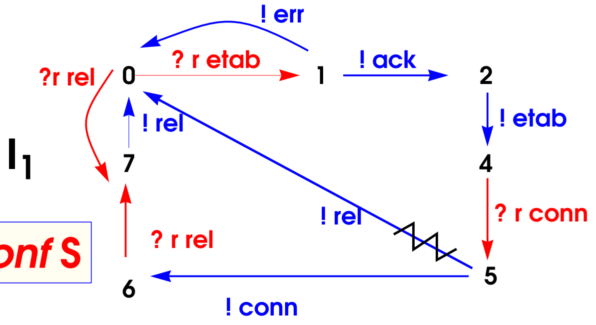
- **Consistency:** weak notion of satisfaction (preorder):

  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 at least a path to the initial state of S.

Conformance

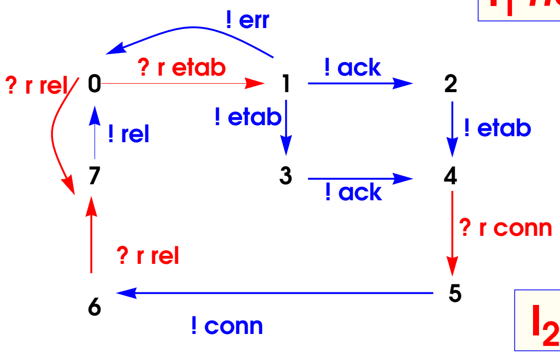
I ioconf S
iff
$\forall \sigma \in \mathbf{traces}(S)$,
$O(I \text{ after } \sigma) \subseteq O(S \text{ after } \sigma)$

Implementations

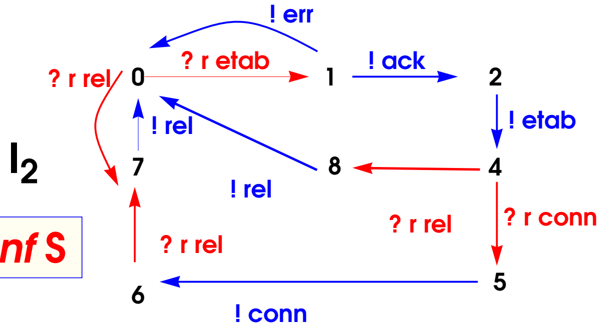


I_1 *not ioconf* S

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:

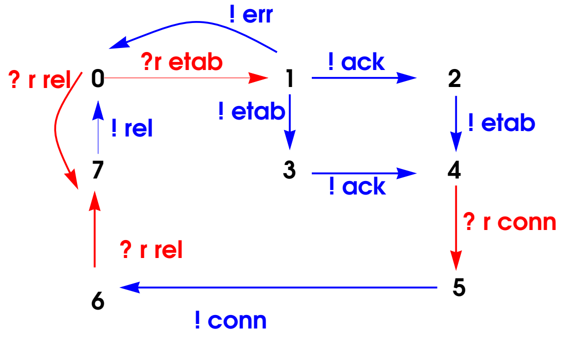
  push :
  - consistency checking
    by testing the reachability of states in (Accept, q^S_{init})).

  pop :
  - synthesis of a DAG from states in (Accept, q^S_{init})
    while assuming controlability,
    decorating some transitions with verdicts
    computing the mirror image
- Second DFS of the resulting DAG
  for the placement of timers operations (Start, Cancel, Timeout).

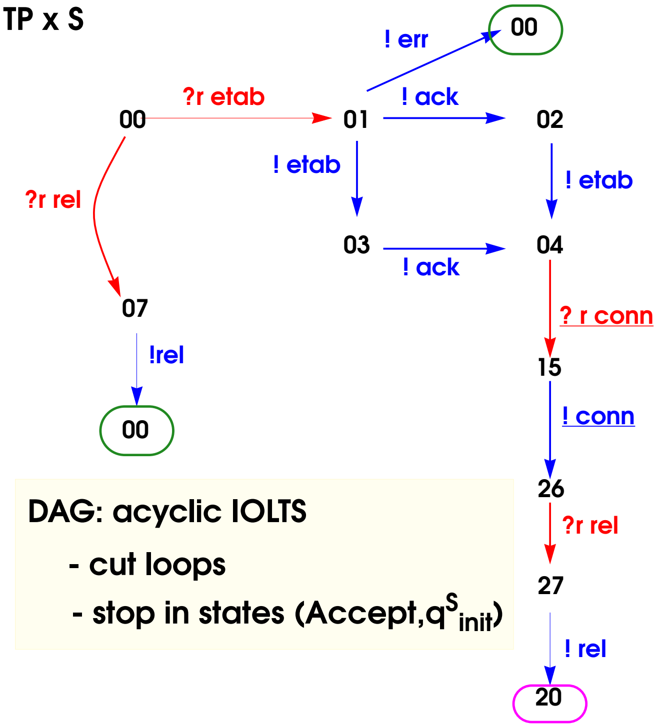
Product : test purpose x specification

Test Purpose TP

Specification S

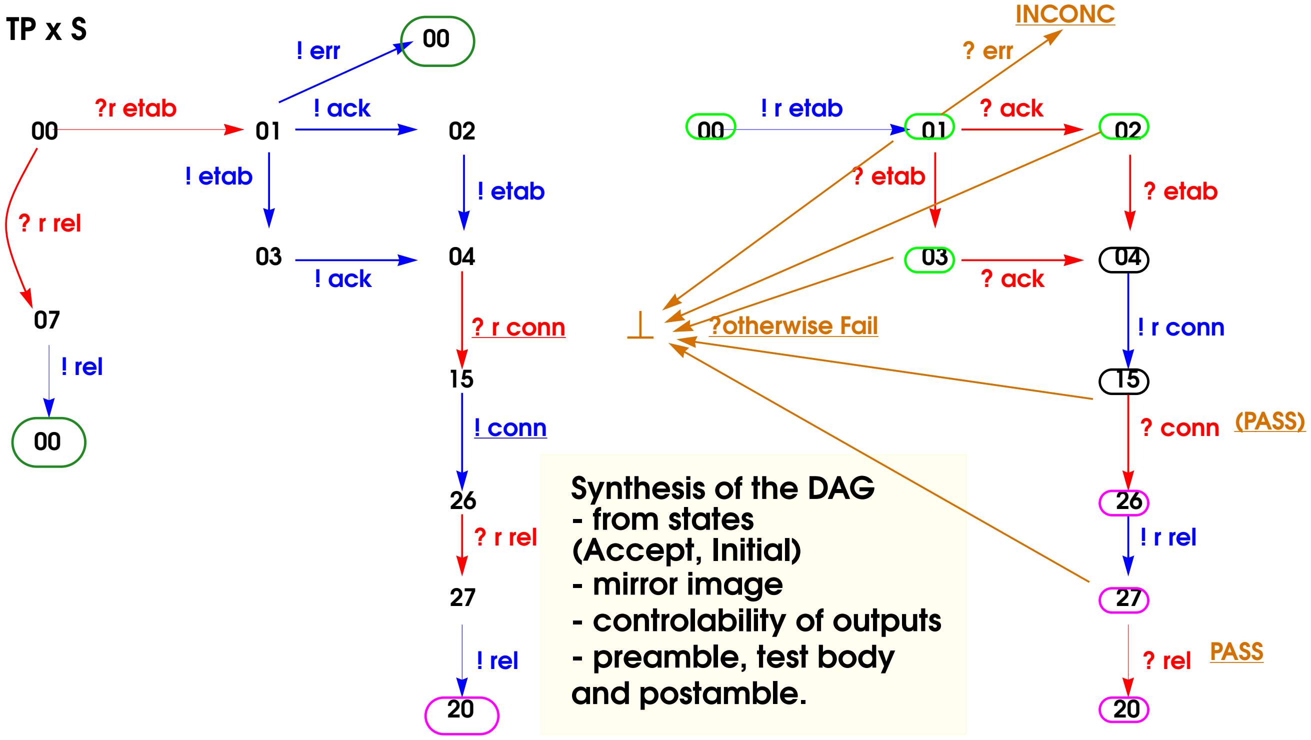


TP x S



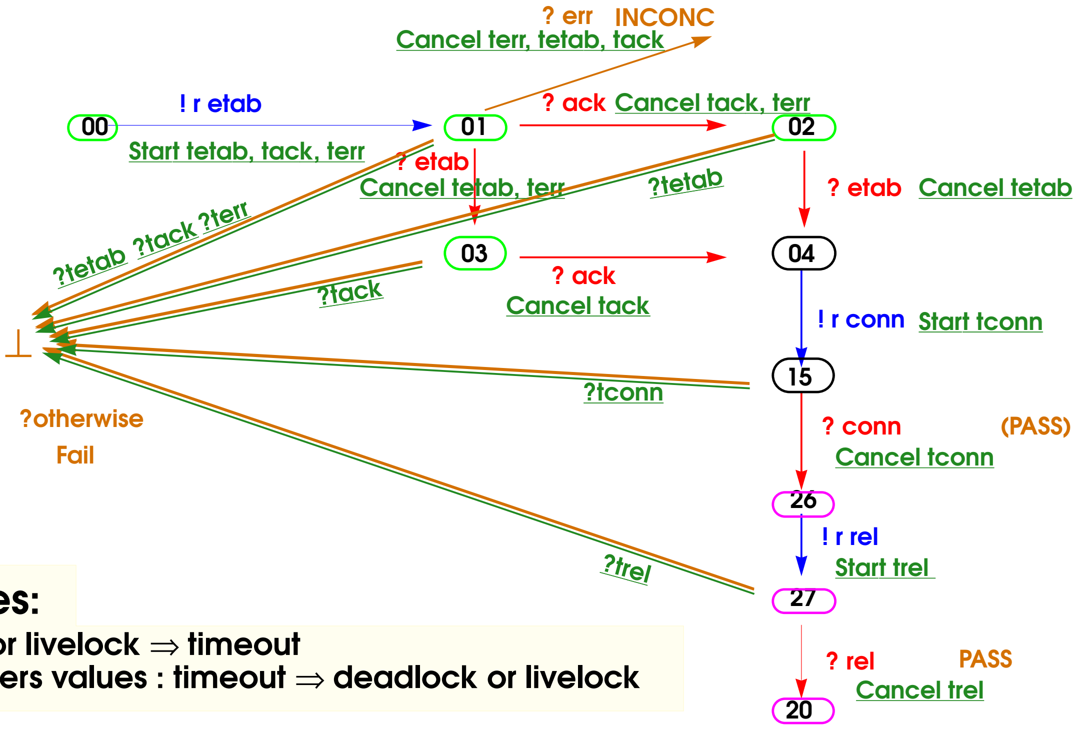
DAG: acyclic IOLTS
- cut loops
- stop in states (Accept,q^S_{init})

Synthesis of a DAG by DFS of TP x S



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

Management of timers



Properties:

deadlock or livelock ⇒ timeout

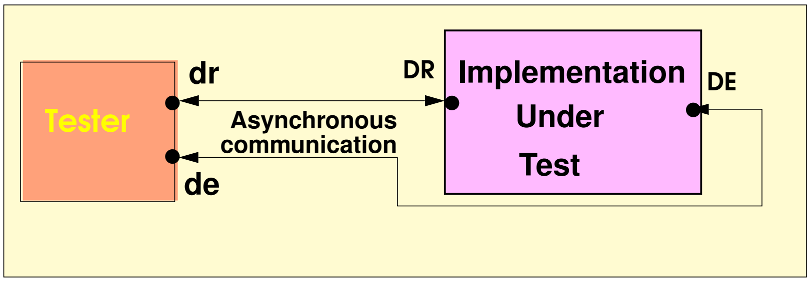
hyp. on timers values : timeout ⇒ deadlock or livelock

Theoretical results

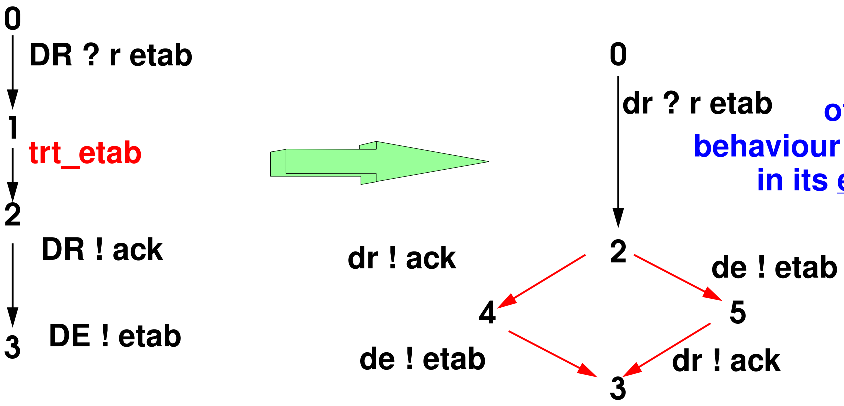
- **Complexity** : linear in time and space / size(**TPxS)**
- **Consistency checking:** the algorithm applied to **TP** and **S** returns true and constructs a test case **TC(TP,S)** iff **TP** is consistent with **S**.
- **Unbiased test cases: hypothesis on timers values**
  TC(TP,S) applied to **I** 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



state graph of the spec.



state graph of the visible behaviour of the spec. in its environment

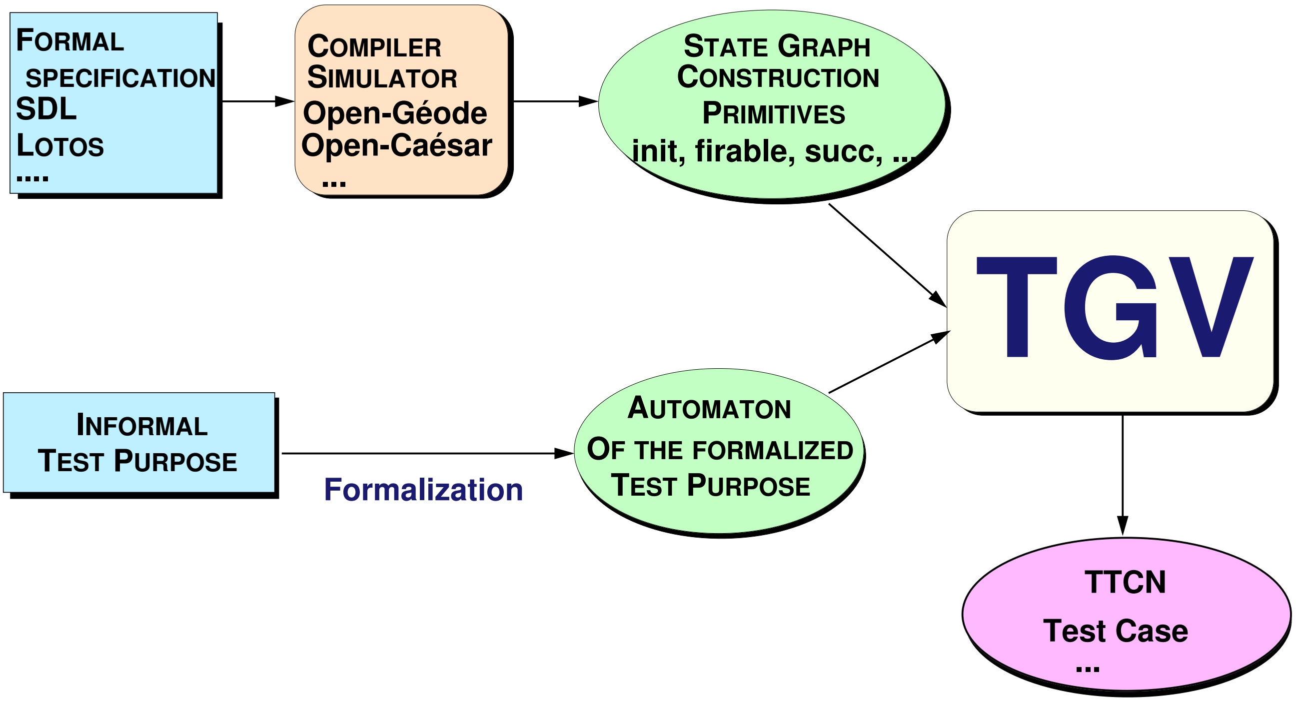
On-the-fly generation

- **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)



FORMAL SPECIFICATION
SDL
LOTOS
....

COMPILER SIMULATOR
Open-Géode
Open-Caésar
...

STATE GRAPH CONSTRUCTION PRIMITIVES
init, firable, succ, ...

TGV

INFORMAL TEST PURPOSE

Formalization

AUTOMATON OF THE FORMALIZED TEST PURPOSE

TTCN
Test Case
...

Experiments

- 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:
  - SSCOP protocol of the ATM in SDL.
  - Memory cach protocol in Lotos (Bull).
  - C.A.N. protocols in car industry in SDL.
- Industrial transfer in preparation

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.