Génération automatique de séquences de tests par la technologie de la vérification

J.-C. Fernandez (Spectre/Vérimag/INRIA-CNRS),
C. Jard (Pampa/Irisa/CNRS),
T. Jéron (Pampa/Irisa/INRIA),
L. Nedelka (Pampa/Irisa/INRIA),
G. Viho (Pampa/Irisa/IFSIC)

Using on the fly verification techniques for the generation of test suites, Computer Aided Verification, CAV 96, New Brunswick(USA).

An experiment in automatic generation of test suites for protocols with verification technology. Science of Computer Programming. Special Issue on Industrial Relevant Applications of Formal Analysis Techniques.

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

Le test de conformité

- **Problème:** vérifier qu'une implantation (de protocole) est conforme à sa spécification.
- **Test boîte noire:** implantation connue uniquement par ses interactions avec l'environnement.
- **Pourquoi ?**

Interopérabilité

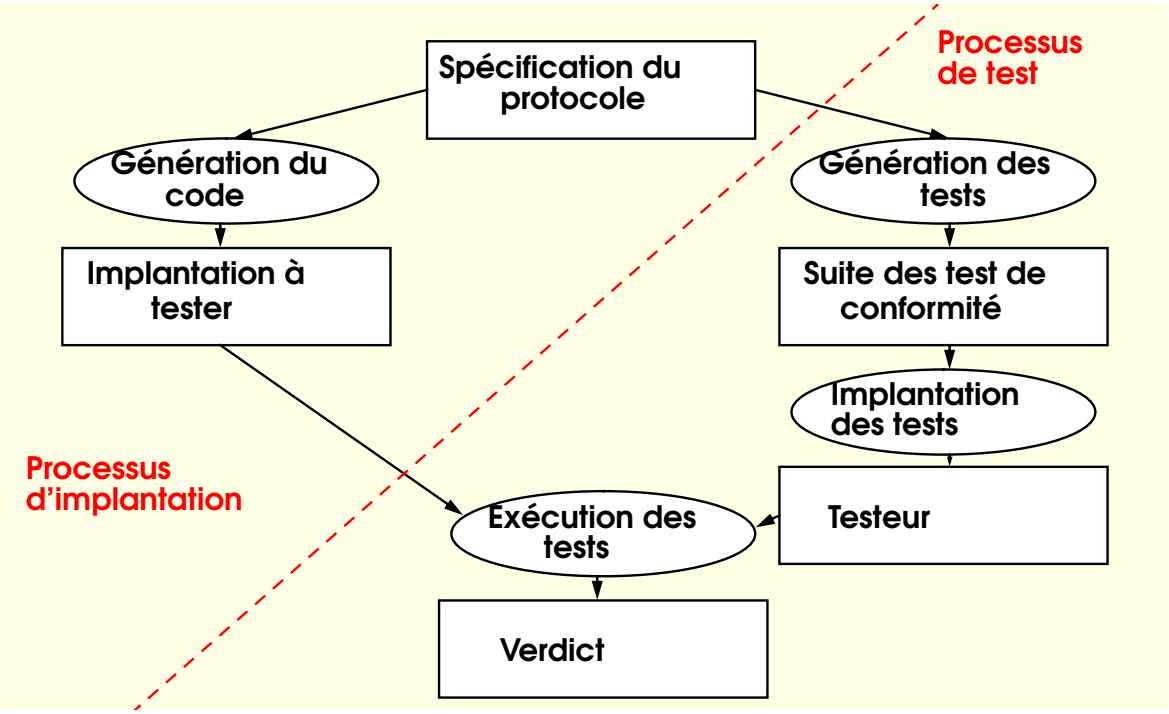
Protocoles normalisés

Implantations conformes

Test de conformité

- **Comment ?** stimuler l'implantation, observer son comportement, en déduire la conformité vis à vis de sa spécification

Le processus de test



Constat sur la génération manuelle

- La génération manuelle de tests est un processus répétitif
- Jusqu'à 20% des tests générés manuellement sont erronés
- La couverture des tests est incertaine
- Difficulté de la maintenance évolutive

⇨ **AUTOMATISER LA GÉNÉRATION DE TESTS À PARTIR DE SPÉCIFICATIONS FORMELLES (LDS, LOTOS, ESTELLE, ...)**

Méthodes formelles et test de conformité

- **Questions**
  - qu'est ce que veut dire conforme ?
  - comment générer des tests ?
  - comment discuter la qualité des tests ?
- **Avantages d'une approche formalisée**
  - définition précise de la conformité et des différents concepts associés au test
  - algorithmes automatiques pour générer des tests
  - validation des tests générés
  - comparaison des méthodes de génération
  - normalisation des algorithmes plutôt que des tests

En pratique...

- On teste l'implantation vis à vis de sa spécification formelle (donnée dans un langage de description formelle), et par rapport à une définition formelle de la conformité.

⇨ *Donner un sens précis aux verdicts des tests (problème de l'oracle)*

- Fournit un complément à l'approche informelle ISO-9646 (Conformance Testing Methodology and Framework)
- Fait l'hypothèse de la correction de la spécification formelle

Constat sur la génération automatique

- méthodes fondées sur les automates à entrée/sortie:

  $s \xrightarrow{i/o} s'$ test: aller en s, appliquer i, observer o, identifier s'

  tour des transitions, UIO, W, DS, monoïde

Gonenc 70, Chow 78, Sabnani et al. 88, Vuong et al. 89, Fujiwara et al. 91, Cavali et al. 93, Chevrier et al. 95.

- méthodes fondées sur les algèbres de processus et la théorie du test

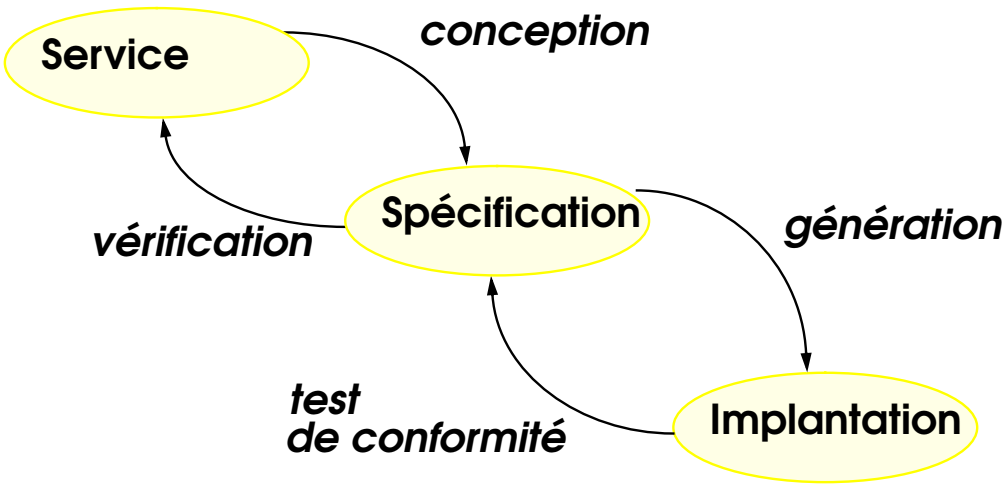
  - systèmes de transition, relation de conformité, testeur canonique.

De Nicola-Henessy 84, Abramsky 87, Brinksma 88, 90, Drira 93, Tretmans 92

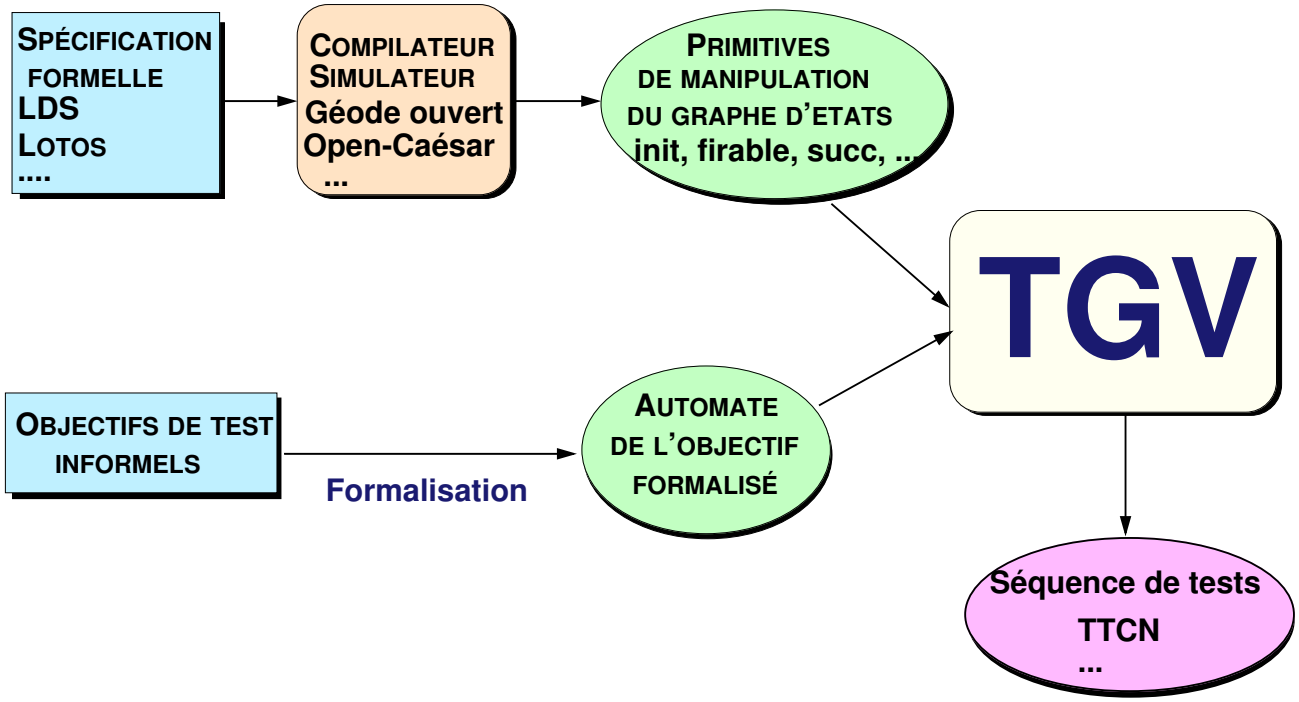
TRÈS PEU D'OUTILS

- complexité des algorithmes/ taille des spécifications
- pertinence des hypothèses
- vraies expériences

Le test et la vérification dans le processus de conception



TGV (IRISA- Verimag)

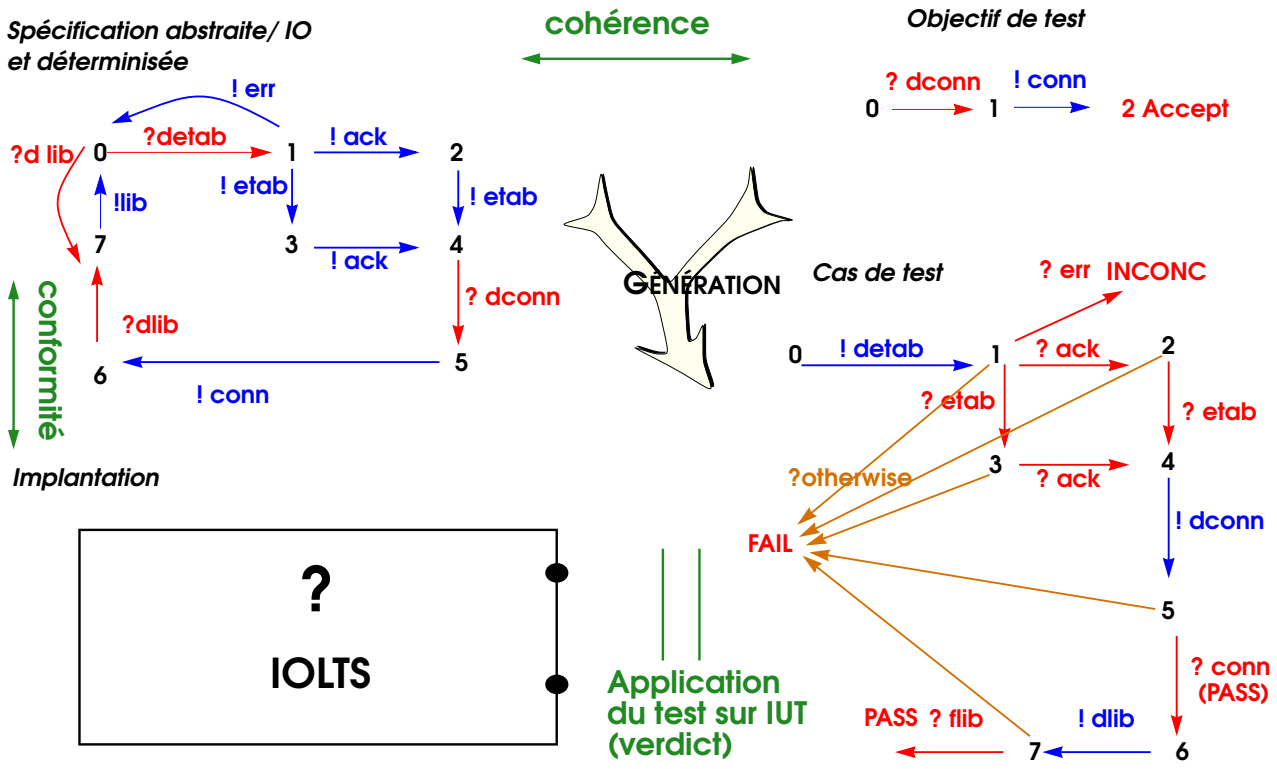


Principes de la génération de test dans TGV

Formalisation de l'activité de génération manuelle

- **Modélisation** de la spécification, de l'implantation, des tests, et des objectifs de test par des **systèmes de transitions à entrée et sorties** (distinction entre actions observables et contrôlables)
- **Définition formelle** d'une **relation de conformité** entre la spécification et ses implantations.
- **Sélection** des tests par les **objectifs de tests**.
- **Correction** des tests vis à vis de la relation de conformité:
  - rejet par le test $\Rightarrow$ non conformité
  - non conformité $\Rightarrow$ on peut produire un test qui peut rejeter l'implantation
- **Efficacité** de la génération $\Rightarrow$ algorithmique fondée sur la **vérification à la volée.**

Modélisation



Système de transitions à entrées/sorties

- Un **IOLTS** est un quadruplet $(Q, A_I, A_O, !, q_{init})$ où
  - Q est un ensemble fini d'états, q_{init} est l'état initial
  - A_I et A_O sont les alphabets d'entrée et de sortie,
  - $A = \{?\} \times A_I \cup \{!\} \times A_O$
  - $! \subseteq Q \times A \times Q$ est la relation de transition.

☞ **Spécification S:** comportement visible modélisé par un IOLTS déterministe.

☞ **Implantation I :** comportement visible modélisé par un IOLTS. $A_I^S \subseteq A_I^I$

☞ **Objectif de test TP:** IOLTS sans cycle, d'alphabets identiques à S,
+ ensemble d'états accepteurs Accept (automate).

☞ **Cas de test TC:** IOLTS sans cycle, déterministe, d'alphabets miroirs de S et I,
+ timers: armements (Start), désarmements (Cancel) et échéances (Timeout)
+ verdicts: FAIL, PASS, (PASS), INCONCLUSIVE
contrôlable: pas de choix entre sorties, ni entre entrée/sortie.

Cohérence entre objectif et spécification

- Un objectif de test décrit (informellement) une propriété d'enchaînement d'interactions de la spécification que l'on veut particulièrement tester.
- Son intention est de sélectionner des comportements parmi les comportements de la spécification.

  Il ne doit donc pas être satisfait par toutes les séquences.
- Cohérence: notion faible de satisfaction:

  Un objectif de test TP est cohérent avec la spécification S si il existe au moins une sous-séquence de S acceptée par TP

Relation de conformité

- Relation liant une implantation I à une spécification S et décrivant formellement quelles sont les implantations I conformes à S.
- Doit tenir compte de la différence entre événements observables et contrôlables.

I *ioconf* S ssi <u>après toute trace de S</u>, les sorties permises pour I doivent être incluses dans celles permises pour S.

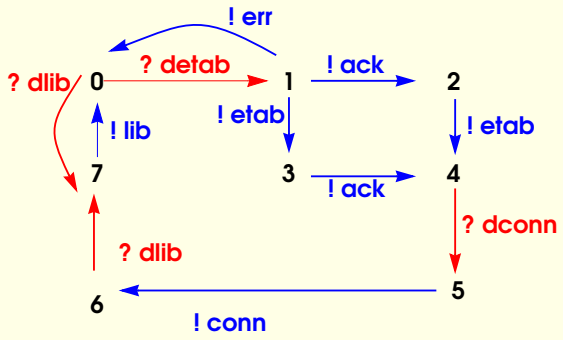
I ioconf S

ssi

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

Conformité : exemples

Spécification S



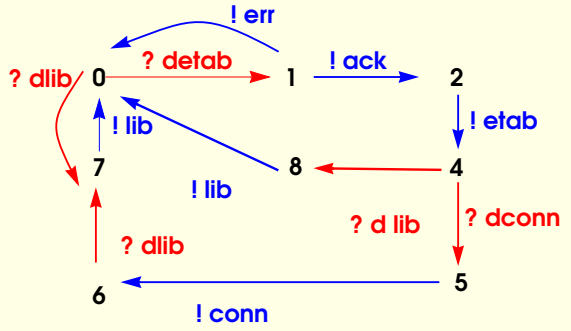
I ioconf S
ssi
$\forall \sigma \in$ traces(S), O(I after σ) $\subseteq$ O(S after σ)

I_1 *ioconf S*

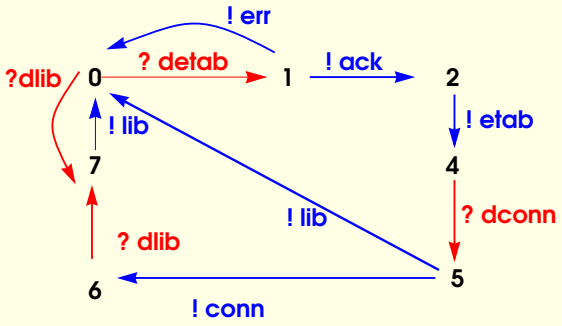
I_2 *not ioconf S*

Implantations

I_1



I_2



Forme des cas de tests à produire

- **Comportement** : ensemble de séquences d'**interactions** entre l'IUT et le testeur, vu du testeur, voulant vérifier la **conformité** de l'IUT à la spécification, par rapport à l'objectif de test.

+ Timers: observation des blocages et livelocks de l'implantation.

- **Verdicts:** associés à certaines interactions avec la signification:
  - **FAIL:** non-conformité.
  - **(PASS):** l'objectif est atteint. Mais une séquence d'actions appelée Postambule doit ramener l'IUT dans l'état initial. Ceci peut conduire à un verdict FAIL. Le verdict (PASS) est donc provisoire.
  - **PASS:** l'objectif a été atteint et l'IUT est dans l'état initial.

  Postambule =la séquence entre le verdict (PASS) et le verdict PASS.
  - **INCONC:** le comportement observé, soit ne permet pas d'atteindre (PASS), soit n'est pas considéré car un cas de test ne peut pas être exhaustif.

Exemple de test TTCN

Nr	Behaviour Description	Constraints Ref	Verdict
1	dr! detab, Start terr, tack, tetab	detab0	
2	dr? etab, Cancel tetab, terr	etab1	
3	dr? ack, Cancel tack		
4	dr! dconn, Start tconn	dconn2	
5	dr? conn, Cancel tconn	conn3	(PASS)
6	dr! dlib, Start tlib	dlib4	
7	? lib, Cancel tlib	lib5	PASS
8	? tlib		FAIL
9	? tconn		FAIL
10	? tack		FAIL
11	dr? ack, Cancel tack, Cancel terr		
12	dr? etab, Cancel tetab	etab1	
13	dr! dconn, Start tconn	dconn2	
14	dr? conn, Cancell tconn	conn3	(PASS)
15	dr! dlib, Start tlib	dlib4	
16	dr? lib, Cancel tlib	lib5	PASS
17	? tlib		FAIL
18	? tconn		FAIL
19	? tetab		FAIL
20	dr? err, Cancel tetab, tack, terr		INCONC
21	? tetab		FAIL
22	? tack		FAIL
23	? terr		FAIL

Principe de l'algorithme

- Algorithme décrit complètement par des règles opérationnelles (à la Plotkin).
- Construction d'un DAG (Graphe Dirigé Acyclique) par parcours en profondeur d'un produit entre S et TP:

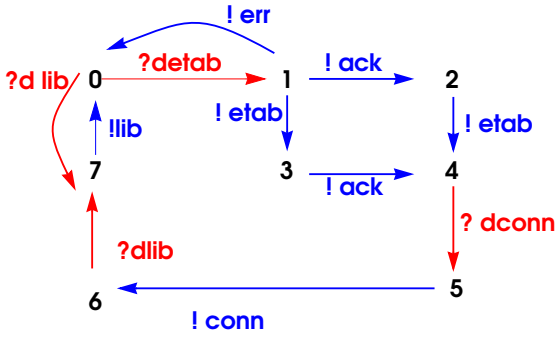
  à la descente:
  - on vérifie la cohérence jusqu'à atteindre des états accepteurs de TP.
  - calcul du postambule par plus court chemin jusqu'à l'état initial de S,

  à la remontée à partir des états accepteurs
  - on synthétise un DAG constituant l'ossature du cas de test TC en assurant la contrôlabilité et en ajoutant des verdicts sur certaines transitions.
- Un deuxième parcours en profondeur du DAG résultant permet de décorer le DAG par la gestion des timers.
- Le DAG est ensuite déplié en arbre pour être traduit au format TTCN.

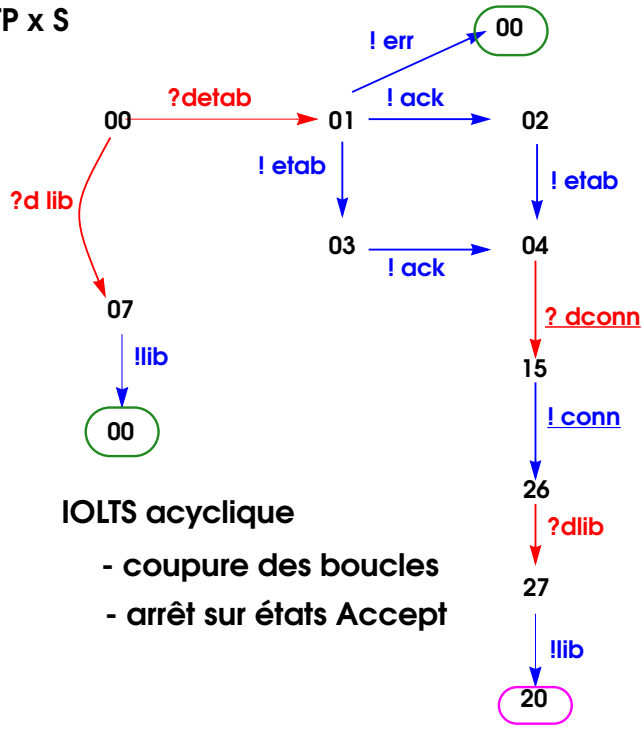
Produit : objectif de test x spécification

Objectif de test TP

Spécification S



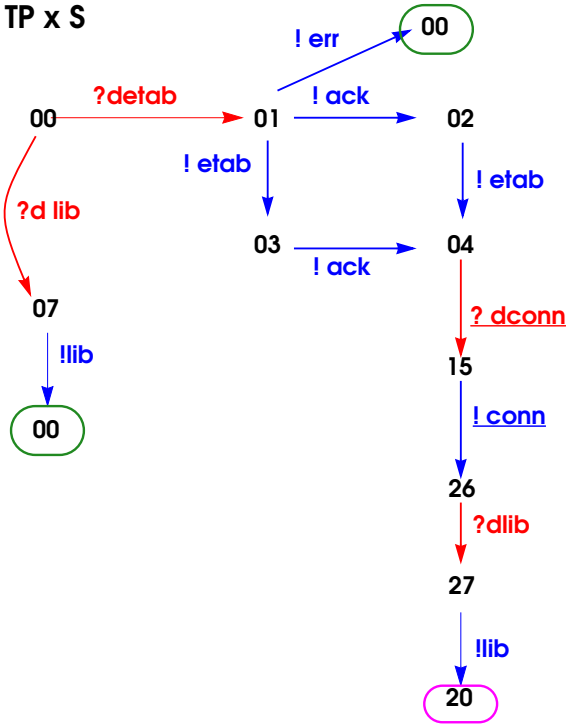
TP x S

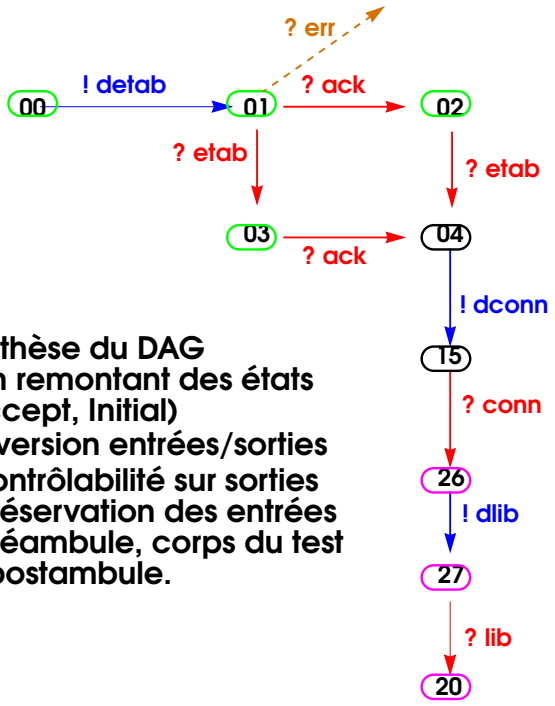


IOLTS acyclique
- coupure des boucles
- arrêt sur états Accept

Synthèse du DAG par DFS de TP x S

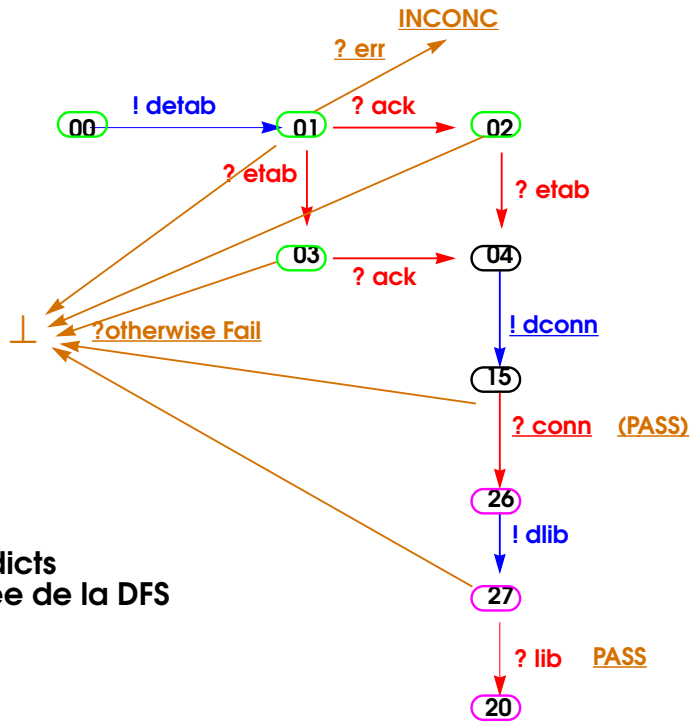
TP x S





Synthèse du DAG
- en remontant des états (Accept, Initial)
- inversion entrées/sorties
- contrôlabilité sur sorties
- préservation des entrées
- préambule, corps du test et postambule.

Verdicts



Décoration par verdicts
pendant la remontée de la DFS

Principe de gestion des timers

assurer la finitude de l'application d'un cas de test sur l'IUT

- **Propriétés attendues:**

  blocage de l'IUT) détection par l'expiration d'un timer,
  Inversement, si les délais des timers sont suffisants,
  expiration de timer) blocage de l'IUT.

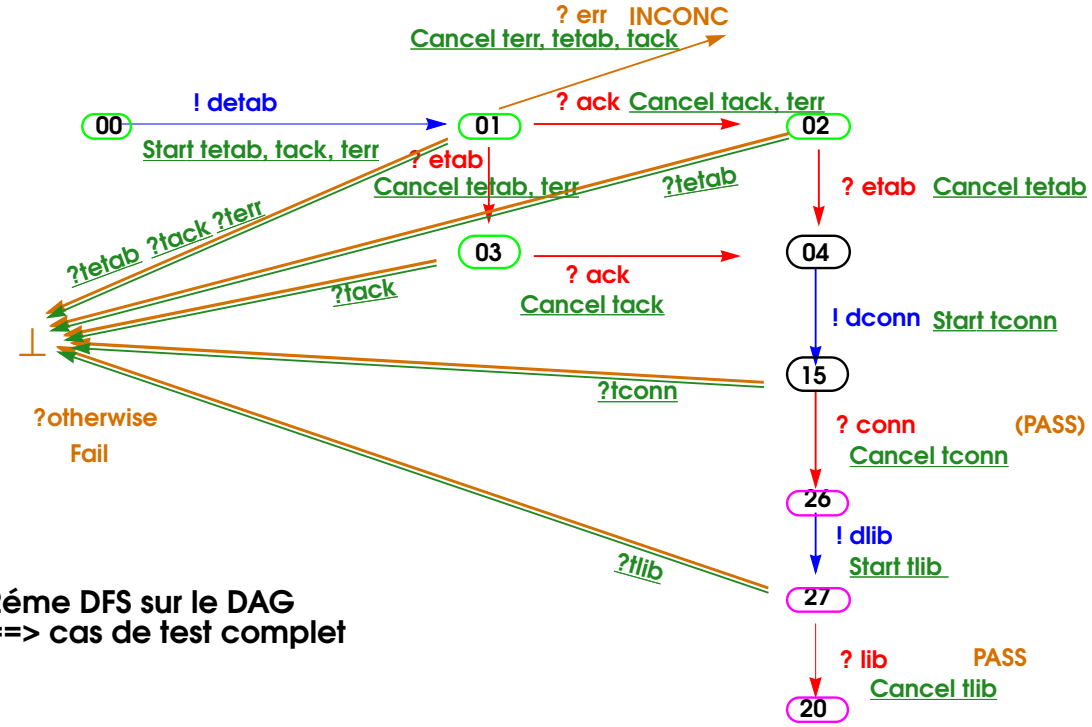
- **Règles de gestion:** à chaque entrée i attendue dans le cas de test correspond un timer t_i. Trois opérations sur t_i :

  - **Start** t_i : doit être fait dans toute transition atteignant le premier état où l'entrée i est possible.
  - **Cancel** t_i : à la réception de i ou quand celle-ci n'est plus possible.

  Attention à la différence entre concurrence et non-déterminisme.

  - **Timeout** t_i : représente l'observation d'un blocage sur l'attente de i. Doit être possible dans les états où i est possible.

Gestion des timers



Résultats théoriques

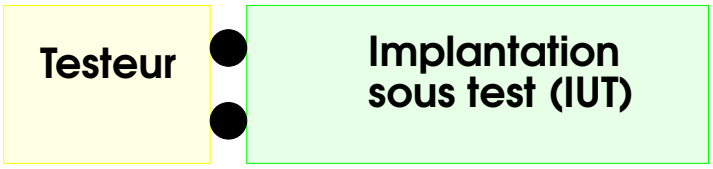
- **Complexité:** linéaire en temps et mémoire par rapport à **TPxS**
- **Vérification de la cohérence :** L'algorithme appliqué à **TP** et **S**, rend vrai et construit un cas de test **TC(TP,S)** si et seulement si **TP** est cohérent avec **S**.

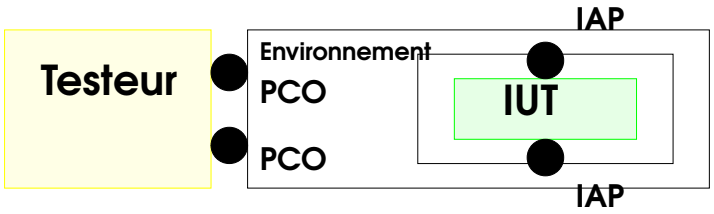
  TC(TP,S) comprend un préambule, un postambule, des verdicts et la gestion des timers.
- **Correction :** si l'application d'un cas de test **TC(TP,S)** généré par l'algorithme sur une implantation **I** produit un verdict **FAIL** alors l'implantation **I** est non conforme à la spécification **S** pour la relation **ioconf**.
- **"Exhaustivité" :** Hypothèse d'équité bornée sur l'implantation **I**.

Pour toute implantation **I** non conforme à sa spécification **S**, il existe un objectif de test **TP** cohérent avec **S** tel qu'un cas de test **TC(TP,S)** généré par l'algorithme et appliqué **n** fois sur **I** produise un verdict **FAIL**.

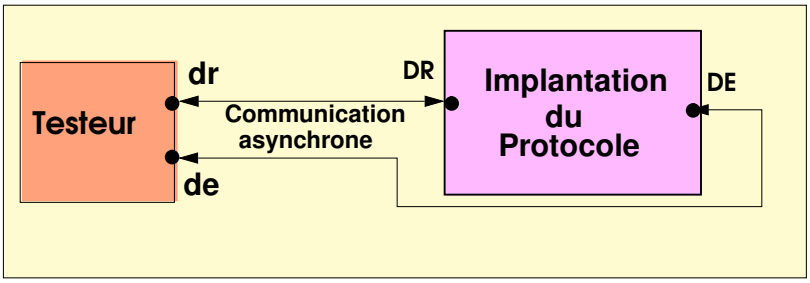
Impact de l'architecture

- Contrôle et observation indirects

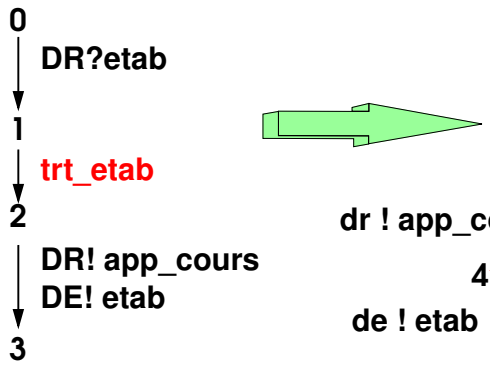


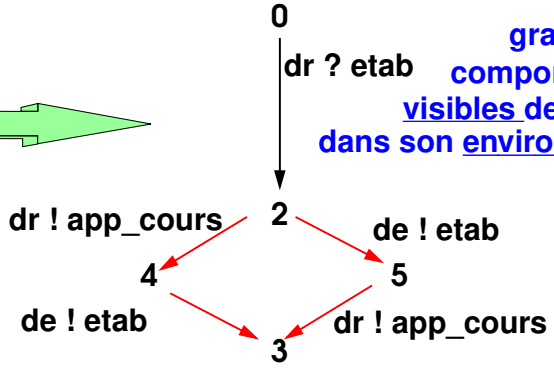


Prise en compte de l'asynchronisme



graphe des comportements de la spec.

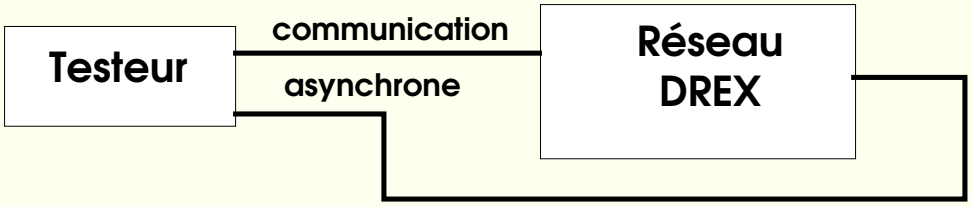




graphe des comportements visibles de la spec. dans son environnement

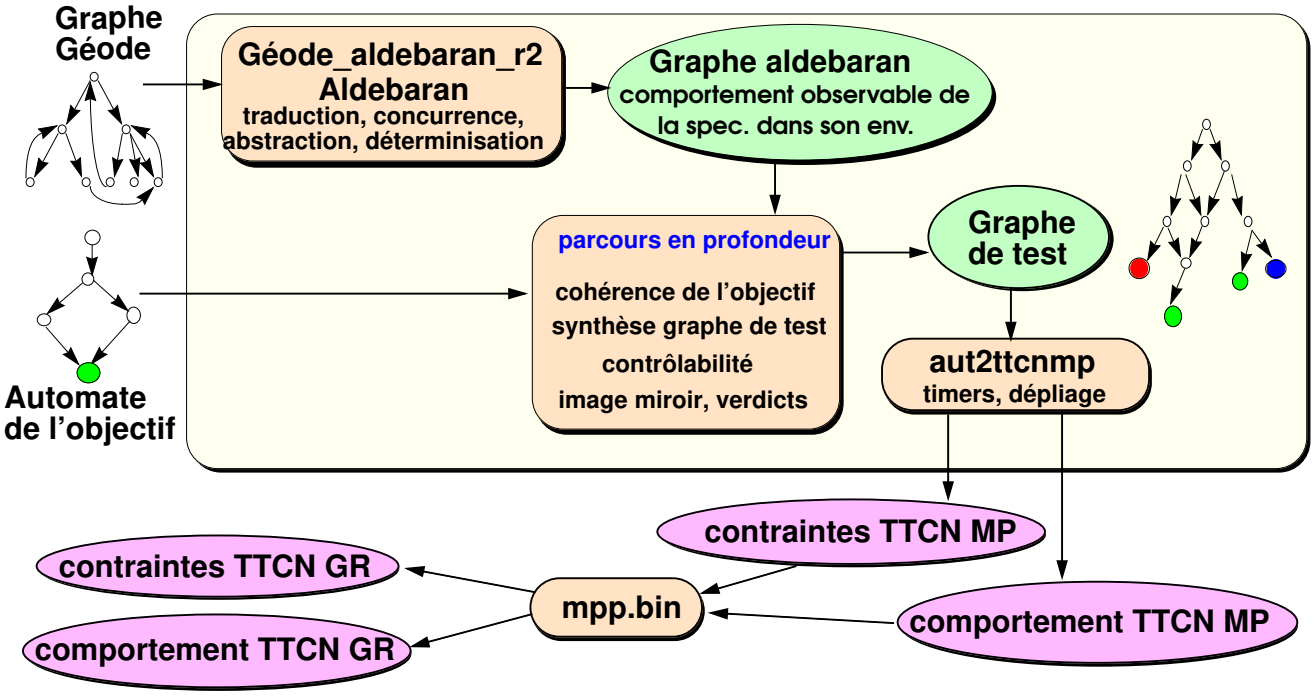
Expérimentation

- **Projet EGT/STSIE/DGA**
  - Etude sur la génération (automatique) de tests de conformité pour les protocoles.
  - Vérilog - Cap-Sesa-Régions - CNET - INRIA - Celar
- **Etude de cas: Protocole DREX**, protocole de signalisation des futurs réseaux militaires à intégration de services.
- **Architecture de test:**



- **Données:** spécification LDS + objectifs + cas de tests manuels
- **But:** générer des tests et comparer avec les tests manuels

1er prototype TGV



Comparaison avec les tests manuels:

meilleure confiance dans les tests générés

- **Verdicts:** fiables car correspondant à la relation de conformité
- **Timers:** gestion automatique “évoluée”
- **Préambules:** pas toujours les plus courts
- **Postambules:**
  - pour nous, pas de postambule après FAIL
  - ni INCONC (à rajouter)
  - plus court chemin.
- **Problèmes de modélisation des objectifs:** inévitable du fait de l’ambiguïté du langage naturel.

Apports de TGV

- **Formalisation du développement automatique de suites de test**
  - utilisation d’une spécification LDS.
  - formalisation des objectifs de tests.
  - formalisation des verdicts PASS, FAIL, INCONCLUSIVE
  - prise en compte de la concurrence due à l’asynchronisme.
  - modélisation de la gestion des timers.
- **Utilisation de l’algorithmique de la vérification**
  - abstraction préservant les entrées/sorties de la spec.
  - calcul des *losanges* modélisant la concurrence.
  - parcours de produit synchrone de graphes.
  - synthèse à la volée du graphe de test.

Génération à la volée

- **Problème:** taille du graphe des comportements de la spécification.
- **But:** produire les tests pendant la construction du graphe.
- **Comment?**
  -) nécessité d'intégrer l'abstraction, la déterminisation dans la construction du graphe.
  -) arrêt à profondeur donnée.

Expérience en cours pour LOTOS dans CADP et à venir pour LDS dans ObjectGEODE.

Perspectives

- **Etude de la commercialisation d'un outil de test autour du simulateur ObjectGEODE (Verilog), intégrant les avantages de TVEDA3 (Cnet) et TGV:**

  génération à la volée, prise en compte de l'environnement de test, gestion des timers, objectifs de tests MSC, ou générés automatiquement par calcul symbolique.
- **Améliorations de TGV:**
  - adaptation pour LOTOS en cours
  - génération à la volée
- **Etudes de cas:**
  - Fast Reservation Protocol (ATM)
  - protocole de cache mémoire (Bull)
- **Test d'interopérabilité:** distribution d'un cas de test sur plusieurs testeurs.

Bibliographie

S. Abramsky. Observational equivalence as a testing equivalence. Theoretical Computer Science, 53(3), 1987.

E. Brinksma, R. Alderden, J. Langerak, R.Van de Lagemaat, and J. Tretmans.A Formal Approach to Conformance Testing. In J. DeMeer, L. Mackert, and W. Effelsberg, editors, Second International Workshop on Protocol Test Systems, p 349-363, North Holland, 1990.

E. Brinskma. A theory for the derivation of tests. In Protocol Specification, Testing and Verification VIII, pages~63--74, North-Holland, 1988.

A.R. Cavalli, S.U. Kim, and Maigron P. Automated Protocol Conformance Test Generation Based on Formal Methods for LOTOS Specifications. In Elsevier Science Publishers B.V., editor, Protocol Test Systems V, p212-222, Montréal, 1993.

C. Chevrier, R. Castanet, B. Le Saëc, R. Casadessus. Génération de séquences de test par calcul de monoïdes. CFIP'95 Ingéniérie des Protocoles. C. Jard et P. Rollin éditeurs. Hermès. Rennes, Mai 95.

T. Chow. Testing software design modelled by finite-state machines. IEEE Transactions on Software Engineering, SE-4(3), may 1978.

M. Clatin, R. Groz, M. Phalippou, and R. Thummel. Two approaches linking a test generation tool with verification techniques. In 8th Int. Workshop on Protocols Test Systems, Evry - FRANCE, September 1995.

R. De Nicola and M. Henessy. Testing equivalences for processes. Theoretical Computer Science, 34:83--133, 1984.

K. Drira, P. Azéma, and F. Vernadat. Refusal graphs for conformance tester generation and simlification: a computational framework. In Protocol Specification, Testing and Verification XIII, IFIP Transactions, North-Holland, 1993.

J.-C. Fernandez, C. Jard, T. J\'eron, and G. Viho. Using on the fly verification techniques for the generation of test suites, Computer Aided Verification, CAV 96, New Brunswick(USA).

J.-C. Fernandez, C. Jard, T. J\'eron, and G. Viho. An experiment in automatic generation of test suites for protocols with verification technology. Science of Computer Programming. Special Issue on Industrial Relevant Applications of Formal Analysis Techniques.

S. Fujiwara, G. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite state models. IEEE Transactions on Software Engineering, 17(6), june 1991.

G. Gonenc. A method for the design of fault-detection experiments. IEEE Transactions on Computing, C-19, june 1970.

Bibliographie (suite)

J. Grabowski, D. Hogrefe, and R. Nahm. Test case generation with test purpose specification by MSCs. In Elsevier Science B.V. (North-Holland), editor, 6th SDL Forum, p253-266, Faergemand, O. and Sarma, A., Darmstadt , 1993.

R. Lai and W. Leung. Industrial and academic protocol testing : the gap and the means of convergence. Computer Networks and ISDN Systems, 27:537--547, 1995.

OSI-Open Systems Interconnection, Information Technology - Open Systems Interconnection Conformance Testing Methodology and Framework - Part 1 : General Concept - part 2 : Abstract Test Suite Specification - part 3 : the Tree and Tabular Combined Notation (TTCN). International Standard ISO/IEC 9646-1/2/3, 1992.

M. Phalippou. Test sequence using Estelle or SDL structure information. In FORTE'94, Berne, October 1994.

M. Phalippou. Relations d'implantations et Hypoth\`eses de Test sur des automates à entrées et sorties. PhD thesis, Université de Bordeaux, France, 1994.

K Sabnani and A. Dahbura. A protocol testing procedure. Computer Network and ISDN Systems, 15(4), 1988.

J. Tretmans. A formal approach to conformance testing. PhD thesis, University of Twente, Enschede, The Netherlands, 1992.

J. Tretmans. Testing Labelled Transition Systems with Inputs and Outputs. Int. Workshop on Protocols Test Systems, Evry - FRANCE, September 1995.

J. Tretmans. Test Generation with Inputs, Outputs and Quiescence. In T. Margaria and B. Steffen, editors, Second International Workshop on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'96, Passau, Germany, March 1996.

S. Vuong, W. Chan, and M. Ito. The UIOv method for protocol test sequence generation. Second International Workshop on Protocol Test Systems, Berlin, October 1989.

L. Verhaard, J. Tretmans, P. Kars, and E. Brinksma. On asychronous testing. In G. Von~Bochman, R. Dssouli, and A. Das, editors, Fifth International Workshop on Protocol Test Systems, IFIP Transactions, North Holland, 1993.