Lot 3.2

Test

Validation du prototype de générateur de séquences de test sur des études de cas

Description :	Cette fourniture fournit des études de cas validant le prototype de générateur de séquences de test pour les systèmes temporisés AVERROES.
Auteur(s) :	Ismail BERRADA, Richard CASTANET, Patrick FÉLIX,
Référence :	AVERROES / Lot 3.2 / Fourniture 3.2.3 / V1.0
Date :	mars 2006
Statut :	validé
Version :	1.0

Réseau National des Technologies Logicielles
Projet subventionné par le Ministère de la Recherche et des Nouvelles Technologies

CRIL Technology, France Télécom R&D, INRIA-Futurs, LaBRI (Univ. de Bordeaux – CNRS), LIX (École Polytechnique, CNRS) LORIA, LRI (Univ. de Paris Sud – CNRS), LSV (ENS de Cachan – CNRS)

Historique

mars 2006	V 0.1	version préliminaire

Table des matières

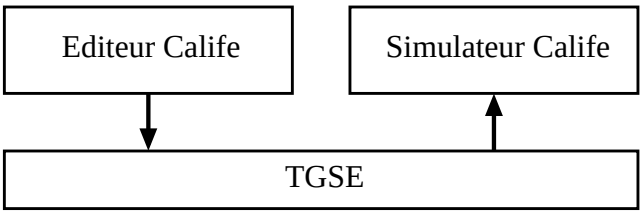

FIG. 1 – Interfaces de TGSE avec Calife.

1 TGSE et la plate-forme logicielle Calife

Calife définit une plate-forme générique (Open Source) permettant d'interfacer des outils de vérification et de génération de test. Un des objectifs du lot 3.2 consiste à intégrer un générateur de séquences de test (TGSE) dans cette plate-forme.

Plate-forme Calife. La plate-forme Calife comporte un éditeur et un simulateur. L'éditeur fournit une interface agréable pour manipuler les différents types d'automates (temporisés, hybrides, étendus,...). Son simulateur permet de simuler graphiquement l'exécution d'un ensemble d'automates. Cette plate-forme supporte deux types d'utilisation : un utilisateur normal qui modélise sa spécification et utilise les outils que la plate-forme fournit pour vérifier ou générer des cas de test, et le mode expert qui permet d'intégrer un outil à la plate-forme.

TGSE sous Calife. Nous avons intégré une partie de TGSE dans calife en définissant les différentes transformations requises. TGSE peut être utilisé en mode graphique à travers Calife. Dans ce cas, la saisie des spécifications se fait à travers l'éditeur Calife qui permet, signalons le, la génération automatique des vecteurs de synchronisation. Il offre le choix d'une synchronisation par rendez-vous, broadcast, la synchronisation binaire d'Uppaal, ou par labels identiques. Ceci pour une topologie statique. L'appel à TGSE se fait à travers l'interface graphique. Calife génère dans ce cas les fichiers d'entrée de TGSE. TGSE produit un cas de test, en format XML selon une DTD Calife, qui pourra être simulé sous Calife. Le schéma de la FIG.1 représente les communications entre Calife et TGSE.

2 Étude de Cas : CSMA/CD

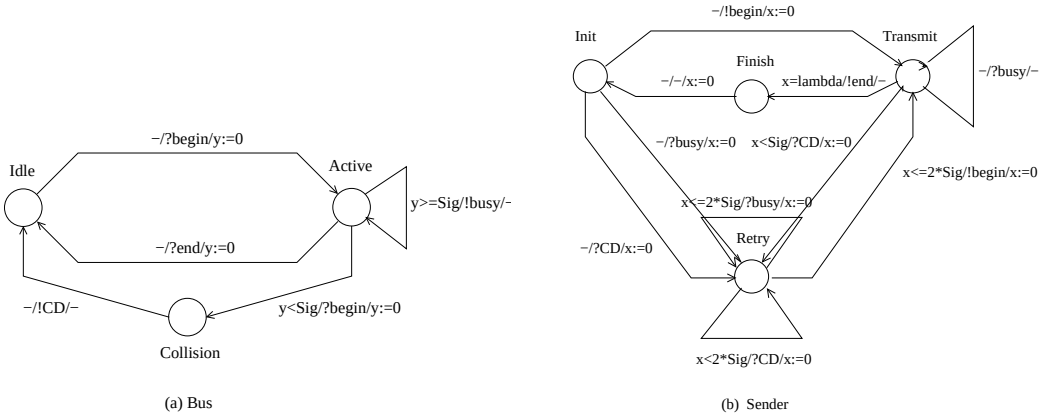

FIG. 2 – Modélisation CSMA/CD.

Nous avons expérimenté TGSE avec le protocole CSMA/CD à un bus, plusieurs émetteurs et un objectif de test consistant à détecter une collision $!CD$ à un instant inférieur à 5 unités de temps. Ceci sous un portable DELL INSPIRON 5100, de processeur Intel P4 (2,4GHZ) et à 256Mo de RAM sous Mandrake 10.0 (TGSE a été aussi testé sous WindowsNT et RedHat). Dans le tableau de la FIG.3, la colonne *Lock* représente le nombre de fois qu'une transition peut figurer dans un cas de test, Taille du TC représente la taille moyenne d'un cas de test généré, Nb Sender correspond au nombre des émetteurs considérés et Temps CPU est le temps moyen de génération. On peut remarquer que la génération avec $Lock = 1$ prend plus de temps. Ceci s'explique par le fait que l'algorithme *gga* atteint des états dont les transitions sont saturées et ainsi il est obligé de dépiler plusieurs fois.

Finalement, bien que le protocole CSMA/CD soit de taille réduite, l'utilisation de plusieurs émetteurs augmente sa complexité. Les résultats obtenus sont très encourageants et des améliorations sont en cours.

Lock	Nb Sender	Taille du TC	Temps CPUs (s)
1	2	3	0.077
1	5	3	0.303
1	10	3	0.621
1	20	3	0.914
10^3	2	18	0.027
10^3	5	55	0.098
10^3	10	79	0.234
10^3	20	130	0.793

FIG. 3 – Expérimentation.