Description d'un outil de génération de cas de test sur des systèmes réactifs temporisés

Salva Sébastien* — **Toussaint Joël****

LIMOS
*** Université d'Auvergne**
Plateaux des Céseaux,
63000 Aubière, France
Salva@iut.u-clermont1.fr

**** Université Blaise Pascal**
Plateaux des Céseaux,
63000 Aubière, France
toussaint@sancy.univ-bpclermont.fr

RÉSUMÉ. *Le test de conformité est devenu une étape essentielle dans le cycle de développement, et permet de vérifier la conformité d'une implantation par rapport à sa spécification. L'activité de test, coûtant plus de 50% des efforts de développement, il est nécessaire de proposer des outils de test offrant une bonne détection d'erreurs avec un coût faible. Dans cet article, nous proposons un prototype d'outil de génération de cas de test sur des systèmes temporisés, dont la complexité est polynômiale. Puis, nous décrivons son application sur deux protocoles réels, le protocole MAP-DSM et le protocole de contrôle audio de Philips.*

ABSTRACT. *Conformance testing is an essential step of the system development cycle and allows to check the conformity of an implementation in comparison with its specification. Generally, testing costs more than 50 % of the overall efforts of development so it is necessary to propose testing tools giving a good error detection with a low cost. In this paper, we present a tool prototype which generates test cases on timed systems modelled with timed automata. Then, we apply it on two real protocols, the MAP-DSM protocol and the Philips audio protocol.*

MOTS-CLÉS : *test de conformité, automate temporisé, cas de test, TTCN*

KEYWORDS: *conformance testing, timed automata, test cases generation, TTCN*

e soumission à *RR003*, le .

1. Introduction

Actuellement, lors du développement de systèmes réactifs temporisés, et de systèmes critiques, le risque de dysfonctionnement de l'implantation augmente suite à de nombreux facteurs tels que la complexité des algorithmes écrits, le nombre d'ingénieurs modélisant ou implantant le système, l'ajout de notions comme le temps ou le multimedia. Malgré cela, ces systèmes doivent rester complètement opérationnels. Ainsi, des solutions ont été proposées pour vérifier leur bon fonctionnement. La validation, basée sur des techniques formelles, est une de ces solutions. Ces techniques sont divisées en deux catégories :

– La Vérification, qui permet de tester la satisfaction de propriétés sur la spécification,

– Le Test de Conformité, qui a pour but de tester le comportement de l'implantation par rapport à la spécification, tout en n'ayant aucune information sur sa structure (l'implantation est vue comme une boite noire). Pour ce faire, des cas de test (suites de propriétés de la spécification) sont donnés ou automatiquement générés pour être ensuite appliqués à l'implantation, par le biais d'une architecture de test, dans le but de détecter d'éventuelles erreurs.

Plusieurs méthodes de test [ENN 98, SPR 01, NIE 01, O.K 00], ainsi que quelques outils basés ou non sur des techniques formelles [NIE 01, BOZ 00, SIN 01], ont été proposés. Or ces outils, soit testent de manière succincte les aspects temporels, soit prennent en compte des modèles de spécification ne permettant pas de représenter tout système. L'activité du test coûtant plus de 50% des efforts de développement, il est essentiel de proposer des méthodes de test offrant une grande capacité de détection d'erreurs et ayant un coût le plus faible possible. Dans cette optique, nous décrivons un prototype d'outil de génération automatique de cas de test sur des spécifications modélisées par les automates temporisés d'Alur et Dill [ALU 94]. Celui-ci, basé sur la méthode [SAL 02], génère des cas de test sous différentes formes et notamment sous forme TTCN (Tree and Tabular Notation [ISO 91]) qui est employé dans le monde industriel. Ce prototype a permis d'obtenir des cas de test sur deux protocoles réel, une partie du protocole MAP-DSM (GSM) et le protocole de contrôle audio de Philips.

L'article est structuré comme suit : en Section 2, nous décrivons un état de l'art du test temporisé. En Section 3, nous définissons les modèles de spécification pris en compte par l'outil, à savoir les automates temporisés et les graphes des régions. La Section 4 donne une vue d'ensemble des techniques formelles utilisées par l'outil de test. Ce dernier est décrit en Section 5. L'étude de cas sur les deux protocoles réels précédents est présentée en Section 6. Nous concluons et donnons quelques perspectives en Section 7.

2. Le test temporisé

Le test de conformité consiste à tester le fonctionnement d'une implantation par rapport à la spécification pour vérifier sa conformité (en procédant à une détection d'erreurs). Pour décrire la nature de la conformité, une relation appelée relation de conformité est définie entre la spécification et l'implantation. Des cas de test sont générés (ou donnés) à partir de la spécification pour démontrer la satisfaction de cette relation, puis sont appliqués sur l'implantation via une architecture de test. Un verdict est alors établi : *pass* l'implantation est conforme à la spécification, *inconclusive* on ne peut répondre, ou *fail* l'implantation contient des erreurs. Voici la description de quelques travaux sur le test temporisés :
Dans [LAU 97], les auteurs ont adapté le concept de testeur canonique au test temporisé. Dans [CLA 97], les auteurs proposent la génération de cas de test à partir de spécifications décrites sous forme de graphe de contraintes. Seuls sont considérés des valeurs minimales et maximale d'horloges permettant l'exécution d'actions de type "entrée/sortie". Dans [ENN 98], la spécification, décrite par un automate temporisé est traduite en un automate proche des machines à états finis (FSM) à partir duquel est appliqué la méthode Wp [FUJ 91], utilisant le concept de caractérisation d'états. Les contraintes temporelles sont placées avec l'etiquette des transitions et sont considérées lors de l'exécution des tests. Dans [O.K 00], la méthode est basée sur une approche orientée objectif de test. Son but est de tester la satisfaction de suites de propriétés sur l'implantation. Un produit synchronisé temporisé est utilisé pour générer des cas de test à partir de la spécification et d'objectifs de test. Dans [SPR 01], les auteurs proposent de traduire l'automate temporisé en un autre automate, le grid automaton. Des cas de tests sont ensuite générés directement sur ce modèle en caractérisant ses états. Une relation de conformité forte (bissimulation) est établie entre la spécification et l'implantation. Des modifications à cette méthode ont été proposées dans [SAL 02]. La méthode décrite dans [NIE 01] considère une sous-classe des automates temporisés, les "event automata" où sur chaque transition une horloge unique est réinitialisée. La construction des cas de test est effectuée en considérant la relation de conformité décrite par Hennessy. Un outil de test a été proposé suite à cette méthode.

L'application des cas de test s'effectue grace à une architecture de test, décrivant les organes permettant de tester et les points d'accès à l'implantation. Dans cet article, nous utilisons l'architecture décrite dans [PET 99].

3. Définitions

Un automate temporisé [ALU 94] est un graphe représentant l'exécution d'un système. Chaque action est décrite par une transition étiquetée par un symbole. Pour représenter le temps dans un automate temporisé, les transitions peuvent contenir des contraintes sur un ensemble horloges.

Définition 3.1 (Contrainte d'horloges) *Soit $C_{\mathcal{A}}$ l'ensemble fini des horloges du système $\mathcal{A}$, et $x_i \in C_{\mathcal{A}}$. Une contrainte d'horloges δ sur x_i est une expression booléenne de la forme $\delta = x_i \Delta c \mid x_i \Delta x_j + c \mid \neg\delta \mid \delta_1 \wedge \delta_2$ où $c \in \mathbb{Q}$ et $\Delta \in \{<, >, =, \leq, \geq\}$.*

Définition 3.2 (Automate temporisé à entrées et à sorties(TIOA)) *Un automate temporisé $\mathcal{A}$ est un tuple $< \Sigma_{\mathcal{A}}, L_{\mathcal{A}}, l^0_{\mathcal{A}}, C_{\mathcal{A}}, E_{\mathcal{A}} >$, tel que : $\Sigma_{\mathcal{A}}$ est un alphabet fini contenant des symboles d'entrées (commençant par " ?") et des symboles de sorties (commençant par " !"). $L_{\mathcal{A}}$ est un ensemble fini d'états, $l^0_{\mathcal{A}}$ est l'état initial, $C_{\mathcal{A}}$ est un ensemble fini d'horloges, $E_{\mathcal{A}} \subseteq L_{\mathcal{A}} \times \Sigma_{\mathcal{A}} \times 2^{C_{\mathcal{A}}} \times \Phi(C_{\mathcal{A}})$ est un ensemble fini de transitions.*
Un tuple $< l, l', a, \lambda, G >$ représente une transition de l'état l vers l'état l' avec le symbole a. L'ensemble $\lambda \in C_{\mathcal{A}}$ correspond aux horloges réinitialisées et G est un ensemble de contraintes d'horloges sur $C_{\mathcal{A}}$.

Définition 3.3 (Valuation d'horloges [ALU 94]) *Une valuation d'horloges sur un ensemble d'horloges $C_{\mathcal{A}}$ est une fonction v qui assigne à chaque horloge $x \in C_{\mathcal{A}}$ une valeur dans $\mathbb{R}+$. Une valuation d'horloges satisfaisant une contrainte d'horloges G, sera noté $v \models G$. Pour $d \in \mathbb{R}+$, $v + d$ dénote la valuation d'horloges qui assigne une valeur $v(x) + d$ à chaque horloge x. Pour $X \subseteq C_{\mathcal{A}}$, $[X \rightarrow d]$ dénote la valuation d'horloges sur $C_{\mathcal{A}}$ qui assigne d à chaque horloge $x \in X$*

Un graphe des régions est une représentation équivalente à un automate temporisé où les contraintes temporelles ne sont pas décrites sur les transitions mais dans les états, par des **régions d'horloges** (polyèdres groupant des valeurs de temps satisfaisant l'exécution d'une même transition).

Définition 3.4 (Graphe des régions [ALU 94]) *Soit $A = (\Sigma_{\mathcal{A}}, S_{\mathcal{A}}, s^0_{\mathcal{A}}, C_{\mathcal{A}}, E_{\mathcal{A}})$ un TIOA. Un graphe des régions de $\mathcal{A}$ est un automate $\mathcal{RA} = (\Sigma_{\mathcal{RA}}, S_{\mathcal{RA}}, s^0_{\mathcal{RA}}, E_{\mathcal{RA}})$ tel que : $\Sigma_{\mathcal{RA}} = \Sigma_{\mathcal{A}} \cup \delta$, où δ représente l'écoulement du temps. $S_{\mathcal{RA}} \subseteq \{\langle s, [v]\rangle \mid s \in S_{\mathcal{A}} \wedge v \in V(C_{\mathcal{A}})\}$. $s^0_{\mathcal{RA}} = \langle s^0_{\mathcal{A}}, [v_0]\rangle$ où $v_0(x) = 0\ \forall x \in C_{\mathcal{A}}$. $\mathcal{RA}$ possède comme transition, $q \xrightarrow{a}_{\mathcal{RA}} q'$, qui part de l'état $q = \langle s, [v]\rangle$ vers $q' = \langle s', [v']\rangle$ avec le symbole a, ssi*
• soit $a \neq \delta\ \exists (s, s', a, \lambda, G) \in E_{\mathcal{A}}$, $d \in \mathbf{R}^+$ tq $(v + d) \models G$, $v' = [\lambda \mapsto 0](v + d)$,
• soit $a = \delta$, $s = s'$ et $\exists d \in \mathbf{R}^+$ tq $v' = v + d$.

Pour simplifier la lecture, nous notons (S, R) un état de graphe des régions, avec S un état de l'automate temporisé et R une région d'horloges. Certains travaux ont montrés comment obtenir des graphes des régions minimaux à partir d'automates temporisés [TRI 96].

4. Méthodologie de test [SAL 02]

La représentation dense du temps au sein des automates temporisés implique un nombre infini de valeurs de temps à tester. Ceci étant impossible à effectuer en pratique, le nombre de cas de test générés doit être limité, tout en offrant une bonne

couverture de l'implantation par les tests. Pour ce faire, l'espace temporel de l'automate temporisé est partitionné de telle sorte que les valeurs de temps, permettant le franchissement d'une transition, sont réunies sous forme d'une région d'horloges. Ceci conduit à la transformation de l'automate temporisé en graphe des régions minimal.
Ensuite, nous recherchons l'ensemble de caractérisation d'états du graphe des régions, qui permet la distinction de chaque paire d'états d'un point de vue comportemental et temporel. Cet ensemble de caractérisation permet de définir une relation de conformité entre la spécification et l'implantation correspondant à une bissimulation [SPR 01]. A partir de ce dernier, deux ensembles de cas de test sont construits, conformément à la méthode Wp[FUJ 91]. Le premier ensemble Π_1 permet de montrer que les états du graphe des régions existent dans l'implantation. Le deuxième, Π_2, permet de montrer que chaque transition du graphe des régions, étiquetée soit par un symbole d'entrée soit par une symbole de sortie existe aussi dans l'implantation.

La méthode [SAL 02] peut se résumer plus précisément par les étapes suivantes :
Pour générer les cas de test, nous définissons un ensemble particulier d'actions, appelé l'ensemble des actions temporisées :

Définition 4.1 (Action Temporisée) *Soit un graphe des régions $\mathcal{RA} = (\Sigma_{\mathcal{RA}}, S_{\mathcal{RA}}, s^0_{\mathcal{RA}}, E_{\mathcal{RA}})$. Une action temporisée $\langle A, R\rangle$ est un couple où A est un symbole de $\Sigma_{\mathcal{RA}}$ et R une région d'horloges.*
Action temporisée d'entrée : *Une action temporisée d'entrée $\langle ?A, R\rangle$ est composée d'un symbole d'entrée. Elle modélise le fait que le symbole A peut être donnée au système pour toute valeur d'horloges de R.*
Action temporisée de sortie : *Une action temporisée de sortie $\langle !A, R\rangle$ est composée d'un symbole de sortie. Elle modélise le fait que ! peut être observé depuis le système pour toute valeur d'horloges de R.*
$\langle \delta, R\rangle$ modélise un écoulement de temps se produisant dans R.

1) Le TIOA, modélisant la spécification est traduit en graphe des régions minimal en utilisant l'algorithme décrit dans [TRI 96].

2) Dans la suite, nous considérons l'ensemble des états S_{min} du graphe des régions RA tel que $S_{min} = \{s \in S_{\mathcal{RA}} \mid \exists s \xrightarrow{e} s'$ avec $e \neq \delta, s' \in S_{\mathcal{RA}}\}$. En effet, ceux-ci (états puits et états ayant uniquement une transition sortante modélisant un écoulement de temps) ne peuvent être caractérisés.

3) Nous générons l'ensemble de caractérisation W sur S_{min}, tel que $W = \{W_{s_i} \mid s_i \in S_{min}\}$, avec W_{s_i} le plus petit sous-ensemble de W permettant de caractériser s_i par rapport aux autres états de S_{min}. Cet ensemble est composé de séquences d'actions temporisées, qui permettent de distinguer deux à deux, d'un point de vue comportemental et temporel, chaque paire d'états de S_{min}. Considérons une paire d'états $(S, S') \in S^2_{min}$, L'ensemble W permet de distinguer S de S' ssi il existe deux ensembles $Obs_S \subset W$ et $Obs_{S'} \subset W$, composés de séquences d'actions temporisées complètement exécutables à partir de S et à partir de S' (séquences composées d'actions temporisées qui peuvent être toutes successivement exécutées), tels qu'au moins

une séquence de Obs_S n'appartienne pas à Obs'_S. Une définition de la caractérisation temporisée, ainsi qu'un algorithme de génération sont donnés dans [SAL 02].
L'algorithme peut se résumer ainsi : une première partie consiste à construire un ensemble T contenant les séquences d'actions temporisées du graphe des régions de longueur comprise entre 0 et l en effectuant un parcours en largeur. Si T caractérise un état S, l'algorithme réduit T pour garder uniquement les séquences caractérisant S. L'ensemble obtenu est l'ensemble de caractérisation de S, W_S. Si tous les états ne sont pas caractérisés, on passe à la longueur $l+1$, jusqu'à ce que l soit égal au nombre de transitions du graphe des régions (toutes les transitions du graphe des régions ont été considérées).

4) Deux autres ensembles d'actions temporisées sont générés : l'ensemble des préambules Q composé de séquences d'actions temporisées qui permettent d'atteindre chaque état de S_{min} et l'ensemble de couverture de transition P, composé de séquences d'actions temporisées qui permettent de franchir chaque transition partant des états de S_{min}.

5) Les deux ensembles de cas de test sont finalement générés :
$\prod_1 = P.W = \{p.\sigma \mid p \in P, \sigma \in W\}$. "." représente la concaténation.
$\prod_2 = P/Q \otimes W = \{r.\sigma_i \mid r \in P/Q, s_0 \xrightarrow{r} s_i$, et $\sigma_i \in W_{s_i}$ si $s_i \in S_{min}\} \cup \{r.\langle\delta, R_i\rangle ... \langle\delta, R_n\rangle\sigma_n \mid r \in R, s_0 \xrightarrow{r} s_i$, et $s_i \xrightarrow{\delta} s_i ... (S_{n-1}, R_{n-1}) \xrightarrow{\delta} s_n$, et $\sigma_n \in W_{s_n}$ si $s_i \notin S_{min}\}$.

Execution des cas de test
Nous appliquons les cas de test précédents sur l'implantation, de la manière suivante :
• Soit $\langle ?A, R\rangle$ un couple d'un cas de test, composé d'un symbole d'entrée $?A$. Soit $v_{init} \in R$, la première valeur d'horloges atteinte par les horloges. Nous proposons que le testeur donne "A" à l'implantation *le plus tôt possible et le plus tard possible*, c'est à dire à la première valeur $v_{init} \in R$ atteinte par les horloges, et une seconde fois à la dernière valeur v_{final} que peuvent prendre les horloges dans R. Pour que les horloges puissent prendre comme valeur v_{final}, un écoulement de temps, dénoté $wait$ peut être nécessaire.
• Soit $\langle !A, R\rangle$ un autre couple d'un cas de test , composé par un symbole de sortie $!A$. Le testeur attend la réception de ce symbole "A" depuis l'implantation, à une valeur de temps de R.
• Autrement, pour un couple $\langle\delta, R\rangle$ d'un cas de test, modélisant un écoulement de temps, le testeur ne procède à aucun test. Il attend de passer au prochain tuple.

5. Fonctionnalités de l'Outil

Nous avons implanté la méthode précédente dans un prototype d'outil appelé TTCG (Timed Test Cases Generation). Une description de son architecture est illustrée en Figure 1. La représentation des régions d'horloges, ainsi que l'implantation de divers opérateurs associés, sont obtenus grâce aux algorithmes implantés dans la librairie Polylib [WIL]. Le prototype d'outil comporte les fonctionnalités suivantes :

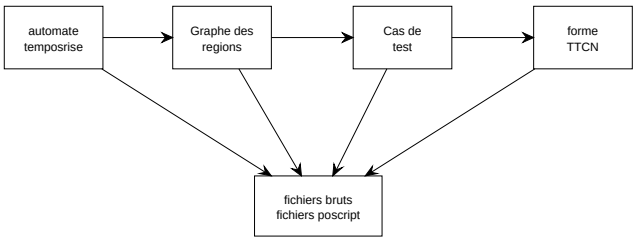

Figure 1. *L'outil de test TTCG*

– **Traduction en graphe des régions :** L'automate temporisé, modélisant la spécification, est transformé en graphe des régions minimal, en utilisant l'outil Kronos.

– **Génération des cas de test :** Les cas de test sont obtenus par la méthode décrite en Section 4. Les cas de test peuvent ensuite être traduits sous forme TTCN [ISO 91]. TTCN (Tree and Tabular Combined Notation) est un language de description de cas de test, employé dans le monde industriel. Chaque cas de test est transformé en arbre (arbre d'exécution représentant chaque exécution sur l'implantation) tout en étant traduit en TTCN. Ceci permet de détailler l'exécution des cas de test, c'est à dire : à quel moment s'effectue l'envoi d'un symbole par le testeur vers l'implantation, dans quelle région d'horloges doit être reçu un symbole par le testeur.

– **Format de sauvegarde :** les spécifications, les ensembles composant les cas de test (ensemble de caractérisation, de préambules et de couverture de transitions) et les cas de test eux-mêmes peuvent être stockés dans un format brut ou par fichiers Postscript.

Terminaison et complexité : *(i)* Dans [TRI 96], les auteurs montrent que l'algorithme de traduction en graphe des régions minimal peut toujours se terminer et que sa complexité est polynômiale sur le nombre d'états et de transition de l'automate temporisé initial. *(ii)* Dans [SAL 02], nous montrons que l'algorithme de génération de l'ensemble de caractérisation d'états se termine et a une complexité proportionnelle à M^2K^2 avec M le nombre d'états du graphe des régions et K le nombre de transitions. *(iii)* Les ensembles des préambules et de couverture de transitions peuvent être facilement construits, avec un coût polynomial [FUJ 91]. Ainsi, les algorithmes se terminent et le coût global de la génération des cas de test est polynomial.

6. Etudes de cas

Le protocole MAP-DSM

Parmi les protocoles utilisés par le protocole GSM (Global system for Mobile communication), neuf protocoles sont groupés dans le MAP (Mobile Application Part). Chacun correspond à une interface spécifique entre deux composants de services au niveau d'une couche du GSM. Le MAP-DSM (Dialog State Machine) gère la coor-

dination entre les services MAP et leurs séquencements (ouverture, fermeture, continuité). Le MAP-DSM permet aussi d'instancier d'autres machines dans le but de démarrer certains services. La figure 2 illustre un partie du protocole MAP-DSM modélisé avec deux horloges. Cette spécification décrit premièrement la requête du service MAP par un utilisateur(?I3).Celui-ci peut invoquer plusieurs requêtes (?I4) qui ont pour but d'invoquer des services (!O3). La spécification décrit de plus la demande de connexion au serveur par un composant de service. Un dialogue peut être accepté puis établi ou être abandonné (!O5 !O9).

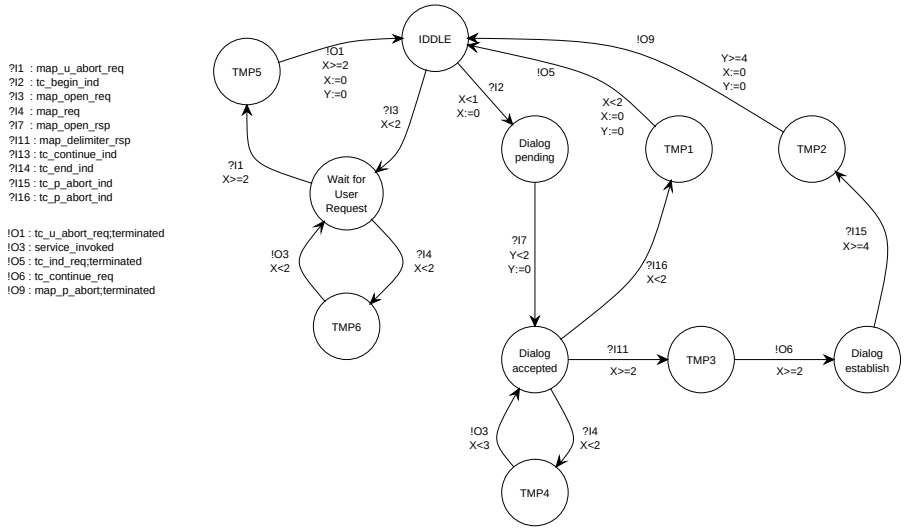

Figure 2. *Une partie du protocole MAP-DSM*

Le tableau suivant décrit les résultats obtenus en applicant la méthode de génération de cas de test sur cette spécification. Nous donnons premièrement le nombre d'états et de transition de l'automate temporisé. Nous faisons de même pour le graphe des régions obtenu. Puis, nous donnons le nombre de cas de test construits ainsi que la somme des longueurs de ces cas. Ces résultats sont obtenus en quelques secondes et demandent environ 5Mo de mémoire.

AT état/transition	GR états/transitions	Nb de cas de test	Lg totale des tests
12	23/31	40	228

Le protocole de contrôle audio de Philips

Ce protocole (Figure 3) est consacré à échanger des informations de contrôle entre des unités audio grand public. Les données sont encodées par le codage Manchester et transmises via un bus. Une station, utilisant ce protocole, instancie deux modules :
•un émetteur, qui reçoit un flot de bits (*in0* pour le bit 0, *in1* pour le bit 1) qu'il encode en utilisant le codage Manchester. Ceci donne une succession de signaux (fronts mon-

tants *up* et descendants *down*) qui sont transmis sur le bus. Si l'émetteur détecte une collision sur le bus, il prévient la couche supérieure en émettant l'action *coll*. Cette détection s'effectue en vérifiant que le Bus est au niveau bas lorsqu'un front descendant est émis. Dans le cas contraire, l'action *isup* est déclenchée.
•un récepteur qui reçoit les signaux (*Vup* lorsqu'il s'agit d'un front montant), qui les décode et qui transmet le flot de données à la couche supérieure (*out0* pour le bit 0, *out1* pour le bit 1).
Deux horloges sont nécessaires pour spécifier des temps limites lors de l'encodage / décodage des valeurs, ces restrictions étant dues à la durée fixe d'un signal sur le bus (1 slot de 888 μs).

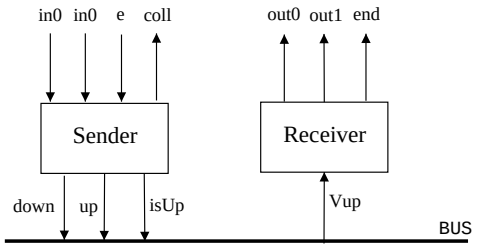

Figure 3. *Le protocole de contrôle audio de Philips*

Voici les résultats obtenus en utilisant l'outil sur la spécification de l'émetteur puis du récepteur. Comme précédemment, ces résultats sont obtenus en quelques secondes et nécessitent qu'environ 6Mo de mémoire.

	AT état/transition	GR état/transition	Nb de cas de test	Lg des tests
Emetteur	16/32	40/66	99	1265
Récepteur	8/13	18/37	70	743

7. Conclusion

Dans cet article, nous avons proposé un prototype d'outil permettant la génération automatique de cas de test sur des automates temporisés. Une traduction des cas de test en TTCN a de plus été prise en compte. Nous avons appliqués cet outil sur deux protocoles réels, le protocole MAP-DSM et le protocole de contrôle audio de Philips. Nous implantons actuellement une seconde méthode basée sur les objectifs de test temporisés. Celle-ci permet de ne tester qu'une suite de propriétés sur l'implantation, ce qui réduit l'exploration de la spécification et réduit les efforts de test.
En amont de la génération des cas de test, nous pourrions évaluer la qualité de test de la spécification. Cette qualité, dépendante de plusieurs critères, permet d'identifier les actions non testables de la spécification et de mesurer une estimation sur le coût de test.
En aval, une autre étape à considérer serait la réduction des cas de test. Ceci aurait pour effet de réduire le coût de l'exécution des cas de test. Il serait intéressant d'exécuter

les précédents cas de test sur des application réelles. Ceci permettrait de définir des architectures de test pratiques, prenant en compte les délais d'acheminement.

8. Bibliographie

[ALU 94] ALUR R., DILL D., « A theory of timed automata », vol. 126, 1994, p. 183-235.

[BOZ 00] BOZGA M., FERNANDEZ J.-C., GHIRVU L., JARD C., JÉRON T., KERBRAT A., MOREL P., MOUNIER L., « Verification and Test Generation for the SSCOP Protocol », *Science of Computer Programming*, vol. 36, 2000, p. 27-52.

[CLA 97] CLARKE D., LEE I., « Automatic Generation of Tests for Timing Constraints from Requirements », *Proceedings of the Third International Workshop on Object-Oriented Real-Time Dependable Systems,* Newport Beach, California, février 1997.

[ENN 98] EN-NOUAARY A., DSSOULI R., KHENDEK F., ELQORTOBI A., « Timed Test Cases Generation Based On State Characterization Technique », *19th IEEE Real Time Systems Symposium (RTSS'98)* Madrid, Spain, 1998.

[FUJ 91] FUJIWARA S., BOCHMANN G. V., KHENDEK F., AMALOU M., GHEDAMSI A., « Test Selection Based on Finite-State Models », vol. 17, n° 6, 1991, p. 591–603.

[ISO 91] ISO, « Conformance Testing Methodology and Framework-Part 1-5 », International Standard n° 9646, 1991, International Organization for Standardization — Information Technology — Open Systems Interconnection, Genève.

[LAU 97] LAURENCOT P., CASTANET R., « Integration of Time in Canonical Testers for Real-Time Systems », *International Workshop on Object-Oriented Real-Time Dependable Systems, California*, IEEE Computer Society Press, 1997.

[NIE 01] NIELSEN B., SKOU A., « Test Generation for Time Critical Systems : Tool and Case Study », *Euromicro01*, , 2001.

[O.K 00] O. KONE R. C., « Test generation for interworking sytems », *Computer communications*, vol. 23, 2000, p. 642-652.

[PET 99] PETITJEAN E., FOUCHAL H., « A Realistic Architecture for Timed Systems », *5th IEEE International Conference on Engineering of Complex Computer Systems, Las Vegas, USA*, 1999, p. 109-118.

[SAL 02] SALVA S., ROLLET A., FOUCHAL H., « Temporal and Behavior Characterization of States in Timed Systems », *ACIS Annnual International Conference on Computer and Information Science, ICIS'02, Seoul, South Korea*, , 2002.

[SIN 01] SINNOTT D. R. O., « Real-Time Systems Development with SDL and Next Generation Validation Tools », *Workshop on Real Time and Embedded Systems, London, UK*, , 2001.

[SPR 01] SPRINGINTVELD J., VAANDRAGER F., D'ARGENIO P. R., « Testing Timed Automata », rapport n° 254, 2001.

[TRI 96] TRIPAKIS S., S.YOVINE, « Analysis of timed systems based on time-abstracting bisimulations », *Proceedings of the 8th International Conference on Computer Aided Verification,* New Brunswick, New Jersey, USA, vol. 1102 de *Lecture Notes in Computer Science*, Springer-Verlag, août 1996.

[WIL] WILDE D. K., « A library for doing polyhedral operations », rapport, IRISA, http ://icps.u-strasbg.fr/PolyLib/.