Un démonstrateur pour le test de conformité de contrôleurs logiques

Julien Provost, Jean-Marc Roussel, Jean-Marc Faure

Un démonstrateur pour le test de conformité de contrôleurs logiques

Julien Provost[1], Jean-Marc Roussel[1], Jean-Marc Faure[1]

[1]LURPA, ENS de Cachan, 61, avenue du Président Wilson, 94235 Cachan cedex - France

{*julien.provost, jean-marc.roussel, jean-marc.faure*}*@lurpa.ens-cachan.fr*

Résumé

Utilisés pour la commande des systèmes critiques, les contrôleurs logiques doivent faire l'objet de tests rigoureux afin de garantir le respect du fonctionnement attendu. La méthode de test proposée dans cette communication s'appuie sur les travaux relatifs au test de machines de Mealy et permet de générer automatiquement une séquence de test exhaustive pour une spécification donnée dans le langage normalisé Grafcet. Un logiciel et un banc de test ont été développés pour mettre en œuvre cette technique de test sur des automates programmables industriels (API).

Mots-clés: *Test de conformité, Test basé sur des modèles, Grafcet, Machine de Mealy, Automate Programmable Industriel.*

1 Introduction

Les systèmes critiques sont aujourd'hui de plus en plus souvent commandés par des contrôleurs logiques, tels que les calculateurs embarqués pour l'automobile ou les API (Automates Programmables Industriels) pour le transport ferroviaire et les centrales électriques. Pour garantir le niveau de sûreté exigé pour ce type de systèmes, ces contrôleurs doivent être testés avant leur mise en fonctionnement, afin de s'assurer de la conformité de leur comportement par rapport au comportement défini par la spécification. Le test de conformité d'un contrôleur logique est effectué en boîte noire et consiste à (cf. figure 1) :

- solliciter le contrôleur à tester par une séquence d'entrée pré-établie ;
- comparer la séquence de sortie observée à la séquence de sortie attendue.

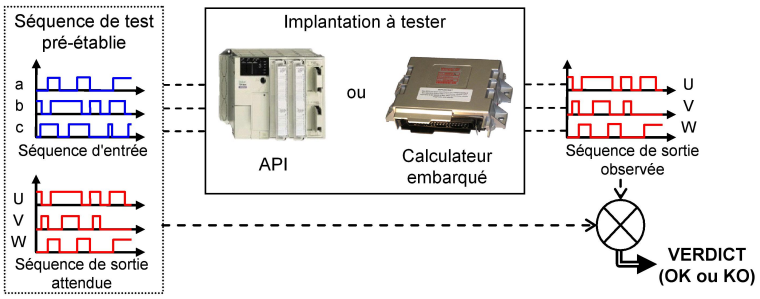

Figure 1 : Principe du test de conformité

La séquence de test sur laquelle repose le test de conformité est donc composée d'une séquence d'entrée et d'une séquence de sortie attendue. Dans la pratique industrielle actuelle, ces séquences sont généralement construites manuellement, activité qui nécessite un travail fastidieux et malheureusement source d'erreurs.

Le test de conformité d'une implantation par rapport à sa spécification est un sujet de recherche pour lequel de nombreux résultats existent. Le but de ces travaux est de construire automatiquement une séquence de test à partir d'une spécification exprimée dans un langage formel tel que les machines à états finis [5, 10], les systèmes de transitions [4, 11] ou, plus récemment, un classe particulière de réseau de Petri [1]. De manière générale, ces résultats proposent une méthode permettant de construire automatiquement une séquence de test à partir d'un modèle de spécification formelle et de donner un verdict à partir de la séquence de sortie observée. Toutefois, dans la limite de nos connaissances, aucun de ces travaux n'a abordé la question de la construction de séquence de test à partir de spécifications exprimées à l'aide de modèles en langage normalisé conçu pour le contrôle/commande. L'objectif de notre travail est de combler cette lacune lorsque la commande est spécifiée en Grafcet [2]. Une méthode permettant d'obtenir, à partir d'un Grafcet décrivant le comportement attendu d'un contrôleur logique, un test exhaustif validant la conformité du comportement du contrôleur logique en fonctionnement par rapport à sa spécification a donc été définie (cf. figure 2). Cette méthode s'appuie sur les travaux effectués sur le test de machines de Mealy. La séquence de test utilisée est conçue pour couvrir la totalité de l'espace d'état défini par la spécification afin de garantir la conformité du comportement.

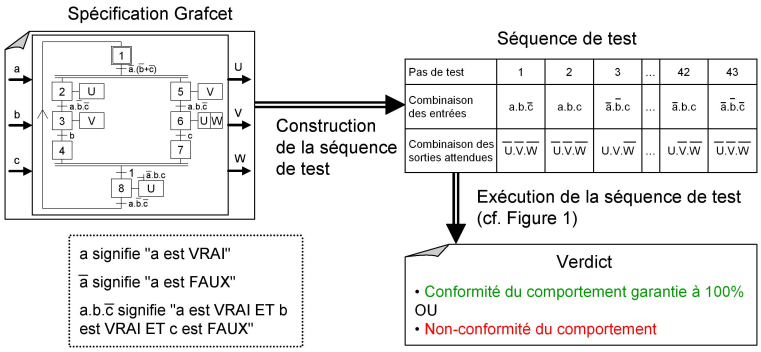

Figure 2 : Objectif de ces travaux

Cette méthode de test est actuellement expérimentée à l'aide d'un banc de test spécifiquement développé pour évaluer la conformité d'APIs. Le logiciel Teloco (TEst of LOgic COntrollers), support de la méthode, ainsi que différents exemples sont disponibles à l'adresse suivante : http://www.lurpa.ens-cachan.fr/isa/teloco/

La section 2 rappelle les principaux résultats en matière de test de conformité de machines de Mealy. La méthode proposée pour la génération automatique de séquence de test à partir d'une spécification Grafcet est ensuite présentée. Les points-clés des différentes phases de cette méthode sont ensuite donnés. La dernière section est consacrée au banc de test développé pour mettre en œuvre un test de conformité sur un API.

2 Test de conformité de machines de Mealy

De nombreux travaux de recherche ont été menés concernant le test de conformité de machines de Mealy. Dans cette section, seuls les concepts principaux seront présentés. Le lecteur pourra se

reporter à la synthèse de ces travaux proposée par Lee et Yannakakis [5] pour tout approfondissement

Une machine de Mealy M (cf. figure 3) est définie par un 6-uplet (I_M,O_M,S_M,s_{InitM},δ_M,λ_M), où :

- I_M et O_M sont respectivement les alphabets d'entrée et de sortie ;
- S_M est l'ensemble des états ;
- $s_{InitM} \in S_M$ est l'état initial ;
- $\delta_M : S_M \times I_M \longrightarrow S_M$ est la fonction de transition ;
- $\lambda_M : S_M \times I_M \longrightarrow O_M$ est la fonction de sortie.

Par définition, une machine de Mealy est déterministe puisque δ_M et λ_M sont définies par des fonctions. Une machine de Mealy est dite complètement spécifiée lorsque les fonctions δ_M et λ_M sont définies pour chaque couple $(s, i) \in S_M \times I_M$. Deux états d'une machine de Mealy, s_i et s_j sont équivalents si pour toute séquence d'entrées, la machine de Mealy produit, à partir de s_i ou de s_j, la même séquence de sorties. Une machine de Mealy est dite minimale si elle ne contient aucune paire d'états équivalents. Deux machines M_1 et M_2 ayant les mêmes alphabets sont dites équivalentes si, pour tout état de M_1 il existe un état équivalent dans M_2 et réciproquement.

Étant données ces définitions, le problème du test de conformité des machines de Mealy peut être décrit ainsi : "soit S une machine connue (la spécification) et I une machine inconnue (l'implantation à tester) qui ne peut être observée et contrôlée que par le biais de ses entrées et sorties, déterminer par un test composé d'une séquence finie d'entrées et de sorties attendues si I est équivalente à S ou non". Pour pouvoir résoudre ce problème, la spécification doit être minimale et fortement connexe (chaque état est accessible à partir de tout autre état). La vérification de la conformité d'une implantation I par rapport à une spécification S revient alors à vérifier qu'aucune des fautes suivantes n'apparaisse lors du test de l'implantation I (cf. figure 3):

- Faute de sortie : depuis un état s, pour une entrée i, I produit la sortie o' au lieu de la sortie attendue o.
- Faute de transfert : depuis un état donné s, après franchissement de la transition étiquetée i/o, l'état d'arrivée de I est s'' au lieu de s'.

La séquence de test, construite à partir de S, doit permettre de détecter ces deux types de fautes, depuis chaque état et pour chaque transition. Chaque test élémentaire correspond donc au test d'une transition $s \xrightarrow{i/o} s'$ de S et comporte trois étapes :

1. Aller à s (synchronisation).
2. Appliquer l'entrée i et vérifier si la sortie o est émise.
3. Vérifier si l'état atteint est bien s' (identification).

Les différents travaux de recherche établis pour le test de conformité ont pour but d'obtenir la plus courte séquence permettant de parcourir S en accord avec les possibilités concrètes d'identification des états de I.

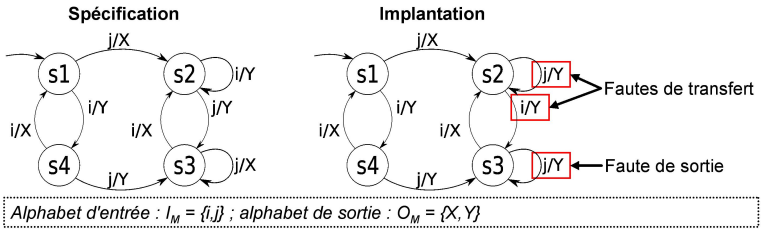

Figure 3 : Fautes de transfert et de sortie dans une machine de Mealy

3 Présentation globale de la méthode

La méthode que nous proposons pour le test de conformité d'un contrôleur logique assurant des fonctions de commande d'un système hautement critique considère que le contrôleur à tester n'est observable qu'au travers ses entrées / sorties. Le test à effectuer doit :

- **Être non invasif**. Aucune sonde ni portion de code ne peut être introduite dans le contrôleur. Il est donc impossible d'obtenir les valeurs des variables internes.
- **Être exhaustif**. La totalité de l'espace d'état de la spécification doit être explorée. Dans la suite de cette communication, la taille de cet espace d'état sera supposée être telle que son parcours exhaustif soit possible. Cette hypothèse est tout à fait raisonnable dans le cas des fonctions de sécurité de systèmes hautement critiques. En effet, ces dernières devant être très réactives, les opérations effectuées sont peu complexes et l'espace d'état de la spécification est calculable en un temps raisonnable. La question du passage à l'échelle ne sera donc pas abordée dans cette communication.

Pour tester de manière exhaustive le comportement d'un contrôleur, il est nécessaire de :

- construire la totalité de l'espace d'état décrit par la spécification ;
- trouver une stratégie permettant de parcourir totalement cet espace d'état ;
- solliciter le contrôleur suivant la stratégie retenue ;
- analyser l'adéquation entre la réponse du contrôleur et celle prévue par la spécification.

Dans le cas du test de conformité d'un contrôleur dont la spécification est donnée en Grafcet (cf. figure 4), il est tout d'abord nécessaire de calculer et de représenter en extension l'espace d'état décrit par la spécification. En effet, contrairement à une machine de Mealy, une spécification Grafcet n'est pas une description où chaque comportement élémentaire est listé explicitement, mais une description dans laquelle chaque comportement élémentaire doit être calculé à partir de règles d'évolutions.

Dans le cas de notre approche, cette représentation en extension, nécessaire pour le test d'un composant physique, se fera sous la forme d'une machine de Mealy. Pour obtenir cette machine de Mealy, nous avons recours à un modèle intermédiaire appelé Graphe des Localités Stables (GLS). Ce graphe est une représentation explicite de tous les états du Grafcet. Il est calculé en recherchant toutes les possibilités d'évolution du modèle Grafcet. Ce GLS est l'équivalent du graphe de marquage d'un réseau de Petri.

Pour des raisons de place et dans un souci de synthèse, les différentes phases de la méthode ne seront pas entièrement détaillées dans cette communication. Les lecteurs intéressés trouveront dans [9] les détails concernant la formalisation du comportement d'une spécification Grafcet, ainsi

que la construction du Graphe des Localités Stables et sa transcription en une machine de Mealy équivalente. Les détails concernant la génération de la séquence de test à partir de la machine de Mealy équivalente au Grafcet peuvent être trouvés dans [7].

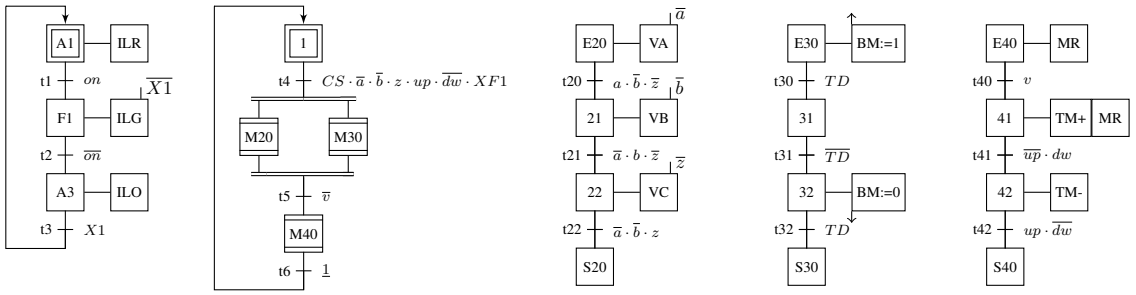

Figure 4 : Exemple de spécification en Grafcet servant de base à un test de conformité

4 Construction du graphe des localités accessibles d'une spécification Grafcet

4.1 Formalisation du comportement d'une spécification Grafcet

Le Grafcet présenté figure 4, établi selon la norme IEC 60848 éditée en 2002 [2], peut être décrit mathématiquement de la façon suivante :

Un Grafcet g est un 4-uplet : $(I_G, O_G, C_G, S_{InitG})$, où :

- I_G est l'ensemble non vide des entrées logiques, (Cardinal de I_G : $|I_G|$).
- O_G est l'ensemble non vide des sorties logiques.
- C_G est l'ensemble des diagrammes d'un Grafcet.
- S_{InitG} est l'ensemble des étapes initiales.

Pour la formalisation du modèle Grafcet, nous distinguerons deux types de diagrammes :

- les diagrammes classiques (les 2 diagrammes de gauche sur la figure 4) ;
- les diagrammes représentant l'expansion d'une macro-étape (les 3 diagrammes de droite sur la figure 4).

Ces diagrammes sont composés d'étapes s, de transitions t et d'actions a. L'ensemble des étapes (étapes d'entrée et de sortie de macro-étape comprises) de ces diagrammes forment l'ensemble des étapes d'un Grafcet. A chaque étape s du Grafcet est associée une variable d'activité $X(s)$; cette variable d'activité peut être utilisée dans les conditions de franchissement des transitions ou d'assignation des sorties.

Une transition t est définie par la donnée de l'ensemble de ses étapes en amont, de l'ensemble de ses étapes en aval, ainsi que de sa condition de franchissement.

Une action a est soit une action continue, soit une action mémorisée.

Une action continue est définie par la donnée de l'étape associée à l'action, de la sortie assignée par l'action, ainsi que de sa condition d'assignation.

Une action mémorisée est définie par la donnée de l'étape associée à l'action, de la sortie affectée par l'action, du type d'affectation effectuée, ainsi que de l'instant où est effectuée cette affectation (à l'activation ou à la désactivation de l'étape).

4.2 Construction du Graphe des Localités Stables (GLS)

Un graphe des localités stables GLS est un 5-uplet $(I_{GLS}, O_{GLS}, L, l_{Init}, Evol)$ où :

- I_{GLS} est l'ensemble des entrées logiques du Grafcet g : $I_{GLS} = I_G(g)$.
- O_{GLS} est l'ensemble des sorties logiques du Grafcet g : $O_{GLS} = O_G(g)$.
- L est l'ensemble des localités l.
- l_{Init} est la localité initiale, $l_{Init} \in L$.
- $Evol$ est l'ensemble des évolutions e.

Une localité stable l est définie par un ensemble d'étapes actives simultanément, un ensemble de sorties émises et une condition de stabilité. Une localité stable est définie par le triplet : $(S_{Act}, O_{Em}, E_{Stab(I_G)})$, où :

- S_{Act} est l'ensemble des étapes actives du Grafcet g pour cette localité ; $(S_{Act} \subset S_G(g))$.
- O_{Em} est l'ensemble des sorties émises du Grafcet g, $(O_{Em} \subset O_G(g))$.
- $E_{Stab(I_G)}$ est la condition de stabilité, définie par une expression booléenne utilisant uniquement les entrées logiques du Grafcet g. Cette expression est vérifiée uniquement pour les combinaisons des entrées pour lesquelles la localité l est stable.

Une évolution e représente une évolution entre deux localités stables. Une évolution peut être définie par le triplet (localité amont,localité aval,condition d'évolution), où la condition d'évolution, définie par une expression booléenne utilisant uniquement les entrées logiques du Grafcet g, est vérifiée uniquement pour les combinaisons des entrées pour lesquelles le GLS évolue de la localité amont à la localité aval.

Un GLS est correctement défini s'il respecte les six propriétés suivantes :

1. Discernabilité des évolutions : aucune évolution ne possède les mêmes localités amont et aval.
2. Les localités amont et aval de chaque évolution sont distinctes, toutes les évolutions représentent un changement de localité : aucune évolution ne boucle sur une même localité.
3. Les évolutions sont déterministes : depuis une localité s'il existe plusieurs évolutions possibles, alors leurs conditions d'évolution sont deux à deux exclusives.
4. Pour chaque localité, aucune combinaison d'entrées ne vérifie à la fois la condition de stabilité de cette localité et une condition d'évolution depuis cette même localité.
5. Le comportement du GLS est complètement défini, pour chaque localité et pour chaque combinaison d'entrées.
6. Le GLS ne contient pas d'évolution fugace.

De plus, la définition d'une localité doit être cohérente avec les règles d'évolutions du Grafcet, c'est-à-dire :

- Lorsque la condition de stabilité d'une localité est vérifiée, alors aucune transition du Grafcet g ne peut être franchie.
- Une sortie assignée par une action continue appartient à l'ensemble des sorties émises si et seulement si elle est associée à une étape active du Grafcet g et que les combinaisons des entrées logiques vérifiant la condition de stabilité de la localité vérifient également la condition d'assignation de la sortie.

Toutes les évolutions d'un GLS sont dues à un changement des valeurs des entrées logiques du Grafcet g. Cependant, ces évolutions peuvent être séparées en deux groupes :

- les évolutions correspondant au franchissement d'une séquence d'ensembles de transitions simultanément franchissables ;
- les évolutions correspondant uniquement à un changement des sorties émises, sans franchissement de transition.

Dans le premier cas, l'ensemble des étapes actives S_{Act} est modifié, alors que dans le second cas, seules les sorties émises O_{Em} et la condition de stabilité $E_{Stab(I_G)}$ sont modifiées.

Le GLS d'un Grafcet est alors construit à partir de la localité initiale en déterminant toutes ces évolutions. Depuis une localité stable, le calcul des conditions d'évolution, des conditions de stabilité et de l'ensemble des sorties émises est effectué par un calcul symbolique sur les expressions booléennes dépendant uniquement des entrées logiques et des variables d'activité des étapes. Cette solution permet d'éviter l'explosion combinatoire et se prête bien à l'étude du Grafcet, pour lequel les conditions de franchissement des transitions et les conditions d'assignation des sorties sont déjà définies par des expressions booléennes.

La détermination des évolutions dues au franchissement d'une séquence d'ensembles de transitions simultanément franchissables depuis une localité stable est effectuée comme suit :

1. Rechercher l'ensemble des transitions franchissables, ainsi que tous les sous-ensembles de transitions simultanément franchissables, depuis la situation (ensemble des étapes actives) S_{Act}
2. Déterminer l'ensemble des situations atteintes lors du franchissement de ces transitions. Réitérer les pas 1 et 2 jusqu'à ce qu'une situation stable soit atteinte.
3. Calculer l'ensemble des sorties émises, composé des sorties suivantes :
   - les sorties assignées par une action continue pour une des étapes actives de la situation stable atteinte ;
   - les sorties affectées par une action mémorisée lors de l'activation ou de la désactivation d'étapes lors de la recherche d'une situation stable.

La détermination des évolutions correspondant uniquement à un changement des sorties émises, sans franchissement de transition, est effectuée en déterminant les combinaisons d'entrées logiques permettant de faire varier les sorties émises sans permettre le franchissement de transition (en ne vérifiant aucune des conditions de franchissement des transitions validées).

Nous rappelons que les détails de la méthode présentée sont disponibles dans [9]. L'outil logiciel Teloco a été utilisé pour traiter le Grafcet illustré figure 4, le GLS de ce Grafcet est composé de 64 localités et 389 évolutions. La construction de ce GLS est effectuée en environ 800ms.

5 Transcription du GLS en une machine de Mealy équivalente et génération de la séquence de test

Un GLS, construit à partir d'un Grafcet, est défini à l'aide d'expressions booléennes associées à chacune de ses évolutions entre deux localités. Une machine de Mealy est un modèle à base d'évènements, c'est-à-dire qu'une transition d'une machine de Mealy est franchie si un évènement apparaît et non si une condition booléenne est vérifiée. Par conséquent, le problème scientifique que l'on cherche à résoudre peut être énoncé ainsi : "comment traduire une machine à états dont les conditions d'évolution sont définies par des expressions booléennes en une machine à états à base d'événements, cela sans perte de sémantique ?".

Comme indiqué en section 2, une machine de Mealy peut être définie par un 6-uplet (I_M,O_M, S_M,s_{InitM},δ_M,λ_M), où I_M et O_M sont les alphabets d'entrée et de sortie. Ces alphabets représentent les ensembles des combinaisons, respectivement d'entrées et de sorties. Ainsi, l'alphabet d'entrée I_M contient $2^{|I_G|}$ éléments et l'alphabet de sortie $2^{|O_G|}$ éléments. A chacun des éléments de l'alphabet est associé un minterme construit à partir des variables, respectivement d'entrées et de sorties. Pour la spécification donnée figure 4, les alphabets d'entrée et de sortie contiennent respectivement 2^9 et 2^{10} éléments ; les mintermes associés sont, par exemple, $m_I(i_{128}) = \overline{CS} \cdot TD \cdot \overline{a} \cdot \overline{b} \cdot \overline{dw} \cdot \overline{on} \cdot \overline{up} \cdot \overline{v} \cdot \overline{z}$ et $m_I(i_{401}) = CS \cdot TD \cdot \overline{a} \cdot \overline{b} \cdot dw \cdot \overline{on} \cdot \overline{up} \cdot \overline{v} \cdot z$

Les états de la machine de Mealy correspondent aux localités du GLS, par conséquent l'état initial de la machine de Mealy correspond à la localité initiale du GLS.

La fonction de transition δ_M est définie à partir des conditions d'évolution et de stabilité du GLS : les conditions d'évolution définissent les transitions entre deux états distincts, tandis que les conditions de stabilité définissent les transitions bouclant sur un même état.

La définition de la fonction de sortie λ_M s'appuie le fait que l'ensemble des sorties émises dépende uniquement de la localité active, et non pas de la localité active et d'une combinaison d'entrées.

Les fonctions de transition δ_M et de sortie λ_M sont complètement spécifiées et assurent le déterminisme d'évolution et d'émission des sorties. Le nombre d'états de la machine de Mealy est égal au nombre de localités du GLS ($|L|$), tandis que le nombre de transitions est égal à $|L| \cdot 2^{|I_G|}$.

Localité stable	Condition d'évolution	Séquence de transitions franchies
$\big(${A1,1},{ILR},$[\overline{on}]\big)$	$\overline{dw} \cdot \overline{on} \cdot up$	$\langle${t42},{t6},{t3}$\rangle$
$\big(${F1,1},{},$[(\overline{CS} + a + b + \overline{z} + dw + \overline{up}) \cdot on]\big)$	$(\overline{CS} + a + b + \overline{z}) \cdot \overline{dw} \cdot up \cdot on$	$\langle${t42},{t6},{t3},{t1}$\rangle$
$\big(${F1,E20,E30},{ILG,VA,BM},$[\overline{TD} \cdot \overline{a} \cdot on]\big)$	$CS \cdot \overline{TD} \cdot \overline{a} \cdot \overline{b} \cdot \overline{dw} \cdot on \cdot up \cdot z$	$\langle${t42},{t6},{t3},{t1},{t4}$\rangle$
$\big(${F1,E20,31},{ILG,VA,BM},$[TD \cdot \overline{a} \cdot on]\big)$	$CS \cdot TD \cdot \overline{a} \cdot \overline{b} \cdot \overline{dw} \cdot on \cdot up \cdot z$	$\langle${t42},{t6},{t3},{t1},{t4},{t30}$\rangle$

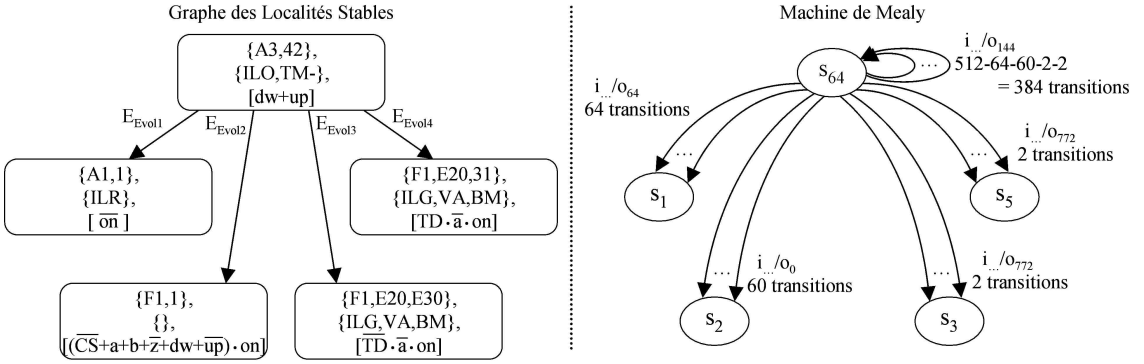

Figure 5 : Représentation des évolutions depuis la localité ({A3,42},{ILR},$[dw + up]$) dans le cas du GLS et de la machine de Mealy pour l'exemple donné figure 4

Pour l'exemple de la figure 4, le GLS contient 64 localités et dépend de 9 entrées logiques, la machine de Mealy équivalente contient donc 64 états et 32768 transitions. Le nombre d'évolutions du GLS (389 évolutions) n'a aucune influence sur la taille de la machine de Mealy (cf. figure 5).

Pour le Grafcet proposé sur la figure 4, la séquence de test est obtenue par la méthode du tour de transition [6] et comporte 73528 pas de test, permettant de tester chacune des 32768 transitions de la machine de Mealy. Cette séquence est obtenue en environ une seconde avec l'outil Teloco ; l'exécution de cette séquence sur le banc de test présenté dans la section suivante dure environ une heure, à raison d'un pas de test toutes les 50 ms.

6 Présentation du banc de test

Afin de pouvoir juger de l'applicabilité de la méthode proposée pour des contrôleurs logiques réels, un banc de test original a été développé.

Le banc de test utilisé actuellement est construit autour d'un module d'entrées / sorties déportées. Ce module industriel (Schneider Electric Momentum 170ENT11001) dispose de 16 entrées logiques contrôlables et 16 sorties logiques observables à distance à partir d'un simple calculateur via le protocole ModBus TCP/IP. Chaque sortie de ce module d'entrées / sorties déportées est connectée en amont d'une entrée du contrôleur logique à tester tandis que chacune de ses entrées est connectée en aval d'une sortie de ce contrôleur logique (cf. figure 6). Cette solution ouverte offre ainsi le maximum de possibilités pour réaliser physiquement les sollicitations et les observations nécessaires au test de conformité.

Concernant le contrôleur à tester, un API modulaire (Phoenix Contact ILC 170 ETH 2TX) a été retenu, en raison de sa bonne réactivité (temps de cycle pouvant être inférieur à 5 ms) et de la conformité de son atelier de programmation à la norme IEC 61131-3 [3].

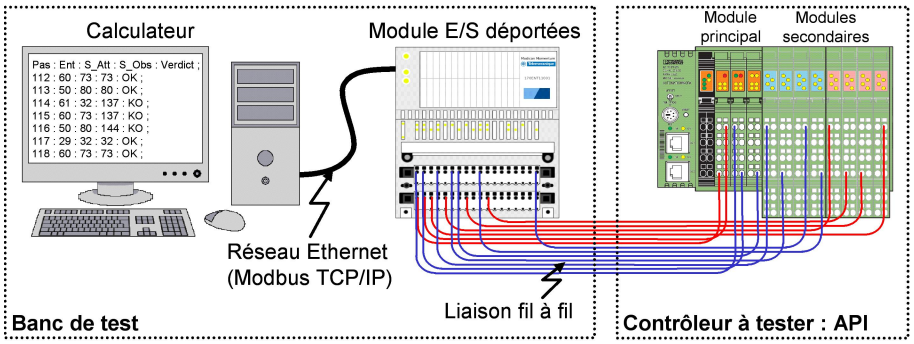

Figure 6 : Description du dispositif expérimental

7 Conclusions et perspectives

Les travaux présentés dans cette communication et illustrés par le démonstrateur montrent qu'il est possible de construire automatiquement une séquence de test permettant d'effectuer un test de conformité exhaustif d'un contrôleur logique à partir d'une spécification non temporisée définie par un Grafcet. Ces résultats permettent ainsi de contribuer à l'amélioration de la sûreté de ces contrôleurs logiques. Les principales contributions de ces travaux sont :

- la définition formelle du comportement d'une spécification Grafcet et du modèle à états finis (GLS) représentant les localités stables et les évolutions de ce Grafcet ;

- une méthode de génération automatique de séquence de test à partir d'une spécification Grafcet. Cette méthode est en expérimentation pour son utilisation dans un contexte industriel.

Les travaux en cours se concentrent sur l'amélioration de la génération des séquences de test. L'approche actuelle, basée sur la méthode du tour de transition, fournit une séquence de test exhaustive de longueur minimale. Dans la plupart des cas, cette séquence est une séquence MVE (à Multiples Variations d'Entrées) : les valeurs de plusieurs entrées logiques peuvent changer d'un pas de test à l'autre. Des expériences menées avec le banc de test développé ont montré qu'il est préférable d'utiliser des séquence de test UVE (à Unique Variation d'Entrée), lors de l'exécution du test du contrôleur logique, afin d'éviter que les changements synchrones des entrées logiques générées par le banc de test soit considérés comme asynchrones par l'implantation à tester. Par conséquent, la génération automatique de séquence de test UVE nécessite d'être approfondie. Ce type de séquence peut être généré à partir de toute spécification vérifiant la propriété d'UVE-testabilité, cette propriété a été définie dans [8]. Lorsque cette propriété n'est pas vérifiée, il est alors nécessaire de construire une séquence de test MVE comprenant le moins de pas de test MVE.

References

[1] von Bochmann G., Jourdan G.-V., *Testing k-Safe Petri Nets.*, LNCS, vol 5826, pp. 33-48, 2009.

[2] IEC 60848, *GRAFCET specification language for sequential function charts*, n 2, 2002.

[3] IEC 61131-3, *Programmable controllers - Part 3 : Programming languages*, 1993.

[4] Jéron T., *Contribution à la génération automatique de tests pour les systèmes réactifs*, Habilitation à diriger des recherches, Université de Rennes 1, 2004.

[5] Lee D., Yannakakis M., *Principles and methods of testing finite state machines - A survey*, Proceedings of the IEEE, vol 84, n 8, pp. 1090-1123, 1996.

[6] Naito S., Tsunoyama M., *Fault Detection for Sequential Machines by Transitions Tours*, Proceedings of the IEEE Fault Tolerant Computer Symposium, pp. 238-243, 1981.

[7] Provost J., Roussel J.-M., Faure J.-M. *Test sequence construction from SFC specification*, Proceedings of DCDS 2009, pp. 341-346, 2009.

[8] Provost J., Roussel J.-M., Faure J.-M. *SIC-testability of sequential logic controllers*, Proceedings of WODES 2010, pp. 203-208, 2010.

[9] Provost J., Roussel J.-M., Faure J.-M. *Translating Grafcet specifications into Mealy machines for conformance test purposes*, Control Engineering Practice, doi:10.1016/j.conengprac.2010.10.001, à paraître.

[10] da Silva Simão A.,Petrenko A., Yevtushenko N., *Generating Reduced Tests for FSMs with Extra States*, LNCS, vol 5826, pp. 129-145, 2009.

[11] Tretmans J. *Model Based Testing with Labelled Transition Systems*, LNCS, vol 4949, pp. 1-38, 2008.