HAL
open science

# Construction d'une séquence de test minimale à partir d'une spécification GRAFCET

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

## To cite this version:

Julien Provost, Jean-Marc Roussel, Jean-Marc Faure. Construction d'une séquence de test minimale à partir d'une spécification GRAFCET. 3èmes Journées Doctorales / Journées Nationales MACS (JD-JN-MACS'09), Mar 2009, Angers, France. CDRom papier N°19. hal-00369851

**HAL Id: hal-00369851**

**https://hal.science/hal-00369851**

Submitted on 22 Mar 2009

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Construction d'une séquence de test minimale à partir d'une spécification GRAFCET

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

Laboratoire Universitaire de Recherche en Production Automatisée (LURPA)
ENS Cachan, 61 Avenue du Président Wilson, 94235 Cachan Cedex, France

{provost,roussel,faure}@lurpa.ens-cachan.fr

*Résumé*—**Cette communication est relative au test de conformité de systèmes électroniques micro-programmés spécifiés en Grafcet. La méthode proposée permet de construire automatiquement une séquence de test de longueur minimale permettant d'effectuer le test exhaustif du comportement du système étudié. Cette méthode tire parti de précédents résultats sur la construction d'une représentation d'un grafcet par une machine à états, et sur le test de machines de Mealy. Cette contribution sera illustrée par un exemple simple.**

*Mots-clés*— **Test de conformité, Test piloté par le modèle, Grafcet, Machine de Mealy, Contrôleurs logiques.**

## I. Introduction

Les systèmes critiques sont de plus en plus souvent contrôlés par des SEM (Systèmes Electroniques Micro-programmés), tels que les calculateurs embarqués pour l'automobile ou les API (Automates Programmables Industriels) pour le transport ferroviaire et les centrales électriques. Afin de garantir le niveau de sûreté exigé, le comportement de ces SEM doit être testé le plus rigoureusement possible. Les travaux décrits dans cette communication ont été réalisés dans le cadre du projet de recherche TESTEC (TEst de Systèmes Temps réel Embarqués Critiques) financé par l'ANR et dont l'un des partenaires industriels est EDF R&D. Pour les systèmes considérés dans ce projet, le test doit être exhaustif et non-invasif.

Le test de conformité d'un SEM consiste à (cf. Fig. 1) :
- solliciter le SEM à tester par une séquence d'entrée ;
- observer sa réponse à cette séquence ;
- comparer la séquence de sortie observée à la séquence de sortie attendue.

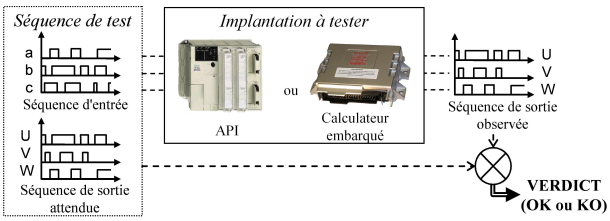
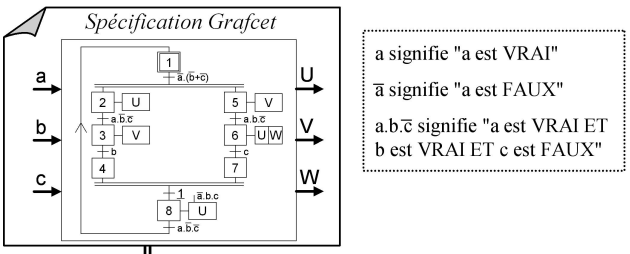



Fig. 1. Principe du test de conformité

Une séquence de test est donc composée de séquences d'entrée et de sortie définies à partir d'une référence. La génération manuelle de ces séquences est une tâche fastidieuse et souvent source d'erreurs. C'est pourquoi le test piloté par le modèle est un sujet de recherche pour lequel de nombreux résultats existent ([1], [2]). 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 de Mealy ou les systèmes de transitions ([2], [3], [4]).

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 conçus pour le contrôle/commande. L'objectif de notre travail est de combler cette lacune lorsque la commande est spécifiée en Grafcet [5]. Nous proposons une méthode pour obtenir, à partir d'un grafcet décrivant le comportement attendu d'un contrôleur logique, une séquence de test permettant de valider si le comportement du SEM est conforme à sa spécification (cf. Fig. 2). Afin de garantir la conformité du comportement, cette séquence doit explorer la totalité de l'espace d'état défini par la spécification. Cette séquence de test doit également être de longueur minimale, afin de réduire la durée d'exécution des tests.




| Pas de test | 1 | 2 | 3 | … | 42 | 43 |
|---|---|---|---|---|---|---|
| Combinaison des entrées | $a.b.\overline{c}$ | a.b.c | $\overline{a}.\overline{b}.c$ | … | $\overline{a}.b.c$ | $\overline{a}.\overline{b}.\overline{c}$ |
| Combinaison des sorties attendues | $\overline{U}.\overline{V}.\overline{W}$ | $\overline{U}.\overline{V}.\overline{W}$ | $U.V.\overline{W}$ | … | $U.\overline{V}.\overline{W}$ | $\overline{U}.\overline{V}.\overline{W}$ |

Fig. 2. Objectif de ces travaux

Les différentes étapes de notre méthode sont présentées dans la section II. Les principes du test de conformité de machines de Mealy sont rappelés dans la section III. La section IV est consacrée à la description de la contribution principale du travail présenté dans cette communication : la transcription de la machine à état équivalente au grafcet initial en une machine de Mealy. Enfin, la section V traite de l'obtention d'une séquence de test de longueur minimale à partir de cette machine de Mealy.

## II. Présentation globale de la méthode

Pour fournir des résultats fiables, le test de conformité d'un SEM contrôlant un système hautement critique doit :

- **Etre effectué en boîte noire**. La structure interne du SEM n'est pas connue et son comportement ne peut être obtenu que par l'observation de ses réponses aux sollicitations d'entrée.
- **Etre non invasif**. Aucune sonde ni portion de code ne peut être introduite dans le système à tester. Il est donc impossible d'obtenir les valeurs des variables internes au cours du test.
- **Etre 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 de fonctions de sécurité de systèmes hautement critiques. Le passage à l'échelle de la méthode de construction de la séquence de test ne sera donc pas abordé dans cette communication.

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

- de construire la totalité de l'espace d'état décrit dans la spécification ;
- de trouver une stratégie permettant de parcourir tout cet espace d'état ;
- d'analyser la réponse du SEM pour savoir si elle est en accord avec le comportement attendu.

Quand la spécification est donnée en Grafcet, il est important de noter les points suivants :

- L'ensemble de l'espace d'état de la spécification n'est pas défini explicitement. Un état correspond à la combinaison de plusieurs étapes actives simultanément, une évolution peut correspondre au franchissement simultané de plusieurs transitions (Problème 1).
- Dans un grafcet, une transition peut être franchissable pour plusieurs combinaisons différentes d'entrées. Pour d'autres combinaisons, aucune transition ne doit être franchie (Problème 2).

Construire la totalité de l'espace d'état d'un grafcet pour décrire son comportement de manière exhaustive pose une réelle difficulté. Tant que ce point n'est pas résolu, le test exhaustif du comportement d'un SEM spécifié en Grafcet est impossible.

Dans cette communication, seuls les systèmes non temporisés sont envisagés. Cela implique que la spécification en Grafcet ne comprend pas de réceptivités ni d'actions dépendantes du temps. Cette limitation est la seule que nous imposons ; toutes les autres structures types du Grafcet (séquence en parallèle, séquences concurrentes, rendez-vous, ...) sont possibles.

La méthode que nous proposons pour obtenir la séquence de test d'un SEM spécifié en Grafcet comporte les 3 phases suivantes :

1. Construction de la Machine à Etats Equivalente (MEE) représentant le comportement du grafcet initial ;

2. Transcription de cette machine à états dotée de conditions de transitions booléennes en une machine de Mealy à base d'évènements ;

3. Construction de la séquence de test de longueur minimale à partir de cette machine de Mealy.

La première phase de la démarche proposée permet d'apporter une solution au problème 1. La seconde phase apporte une solution au problème 2.

Les lecteurs intéressés pourront trouver dans [6] une présentation détaillée de la phase 1 de cette méthode. Nous rappellerons ici les principales caractéristiques de la MEE construite.

- Tout grafcet ne comportant pas d'élément temporisé (réceptivité ou action) peut être transcrit en une machine à états décrivant toutes les évolutions possibles de ce grafcet. Cette machine à états sera notée $MEE$.
- Une MEE ne contient qu'un état actif à la fois, correspondant à la situation courante du grafcet.
- Chaque état d'une MEE est atteignable depuis l'état initial.
- Les conditions de transition d'une MEE sont des expressions booléennes formées à partir des entrées du grafcet initial.
- Les valeurs des sorties d'une MEE ne dépendent que de l'état actif.
- Dans une MEE, il n'y a pas d'évolution fugace (une définition formelle de ce point sera donnée en section IV). Si le grafcet initial comporte des évolutions fugaces, celles-ci sont en effet remplacées par des évolutions non fugaces lors de la construction de la MEE.
- Une MEE peut être représentée graphiquement en utilisant la syntaxe Grafcet. Avec cette représentation, chaque étape représente un état de la MEE. Ainsi, ce grafcet correspondant n'a donc qu'une seule étape active à tout instant.

La figure 3 présente un exemple simple de MEE décrite en Grafcet. C'est sur cet exemple que seront illustrées les phases 2 et 3 de notre approche.

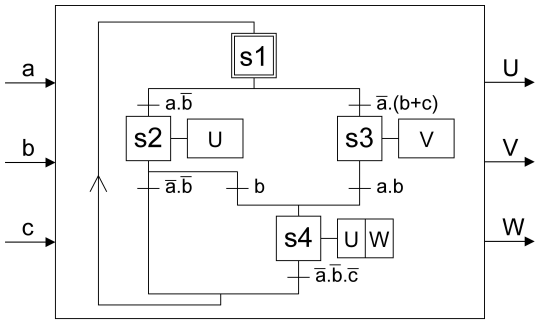



Fig. 3. Exemple d'une MEE

Pour permettre un test non invasif, nous supposerons qu'à chaque étape de la MEE est associée un ensemble différent d'actions, permettant ainsi la connaissance de l'étape active uniquement par observation des sorties.

Avant de détailler la phase 2 de notre méthode de construction de la séquence de test en section IV, il est important de rappeler le principe du test de conformité de machines de Mealy ; ceci est le thème de la section suivante.

## III. Test de conformité de machines de Mealy

Le test de conformité de machines de Mealy a fait très tôt l'objet de nombreux travaux de recherche. Une synthèse

de ces travaux est proposée dans [3]. Nous ne présentons ici qu'un rapide descriptif des fondements de ces techniques de test extrait de [1] et [3].

D'un point de vue mathématique, une machine de Mealy $M$ est un 6-uplet ($I_M$,$O_M$,$S_M$,$s_{initM}$,$\delta_M$,$\lambda_M$) où :

- $I_M$ et $O_M$ sont des ensembles non vides de symboles, respectivement d'entrée et de sortie ;
- $S_M$ est un ensemble non vide d'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, les machines de Mealy sont déterministes puisque $\delta_M$ et $\lambda_M$ sont définies par des fonctions. Il est généralement supposé que ces machines sont complètes, c'est-à-dire que les fonctions $\delta_M$ et $\lambda_M$ sont définies pour tout couple $(s, i) \in S_M \times I_M$.

Une machine de Mealy à 4 états construite sur les alphabets d'entrée $\{a, b\}$ et de sortie $\{X, Y\}$ est donnée Fig. 4.

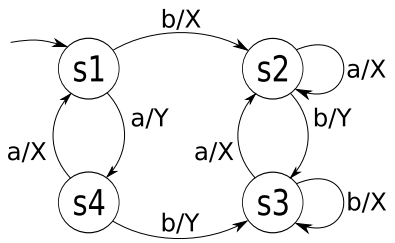



Fig. 4. Exemple d'une machine de Mealy

Dans une machine de Mealy, deux états $s_i$ et $s_j$ sont dits équivalents si pour toute séquence d'entrée ($\sigma \in {I_M}^*$) [1], la machine de Mealy produit la même séquence de sortie :

$$\forall \sigma \in {I_M}^* \, [\lambda_M(s_i, \sigma) = \lambda_M(s_j, \sigma)] \qquad (1)$$

Une machine de Mealy est dite minimale si elle n'a pas de paire d'états équivalents. Deux machines $M_1$ et $M_2$ de mêmes alphabets sont dites équivalentes si pour tout état de $M_1$ il existe un état équivalent dans $M_2$ et vice versa.

En accord avec ces définitions, le problème du test de conformité piloté par le modèle peut être énoncé de la manière suivante :
Soit $S$ une machine connue (la spécification) et $I$ une machine inconnue (l'implémentation à tester) dont on peut seulement observer les entrées/sorties, déterminer par un test (une séquence finie d'entrées/sorties) si $I$ est équivalente à $S$.

Pour que ce problème ait une solution, il est généralement supposé que la spécification est minimale et fortement connexe. L'équivalence entre une implantation $I$ et une spécification $S$ revient alors à vérifier qu'aucune des fautes suivantes n'apparaisse lors du test de $I$ :

- 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 est $s''$ au lieu de $s'$.

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

1. Aller en $s$ (synchronisation).
2. Appliquer l'entrée $i$ et vérifier la sortie $o$.
3. Identifier l'état d'arrivée $s'$ (identification).

Lors de l'exécution de cette séquence se posent les problèmes de synchronisation et d'identification des états de la machine de Mealy. Pour cela, une implantation $I$ peut ou non, disposer de deux fonctions particulières : les fonctions *reset* et *status*. La fonction *reset* permet de ramener la machine de Mealy à son état initial depuis tout état courant, réduisant ainsi les séquences de synchronisation. La fonction *status* simplifie l'identification de l'état courant de la machine par l'émission d'un symbole associé à cet état.

Lorsque la fonction *status* existe, la séquence de test peut être obtenue simplement en appliquant la méthode du tour de transition [7]. Cette méthode consiste à déterminer la plus courte séquence de symboles d'entrée permettant de traverser au moins une fois chaque arc du graphe correspondant à la machine de Mealy. La requête *status* est alors émise entre chaque pas de la séquence de test, permettant ainsi de déterminer l'état courant de la machine $I$ testée. La simplicité de mise en oeuvre et l'existence d'outils mathématiques permettant son optimisation sont les principaux avantages de cette méthode.

## IV. Transcription d'une Machine à Etat Equivalente en une machine de Mealy

Cette section vise à définir les règles de transcription de la MEE, décrite dans la syntaxe Grafcet, en une machine de Mealy à base d'événements (phase 2 de la méthode proposée section II). Avant tout, le comportement de la machine de Mealy obtenue devra être identique à celui de la MEE initiale. Il est important de rappeler qu'aux transitions d'une MEE sont associées des conditions booléennes. A l'inverse, une machine de Mealy est un modèle à base d'événements. Par conséquent, le problème scientifique posé est la transcription d'une machine à états finie à base de conditions booléennes en une machine à base d'évènements, cela sans perte de sémantique.

### A. Définition formelle d'une Machine à Etats Equivalente

Une MEE peut être décrite par le 6-uplet :
$MEE = (V_I, V_O, S_{MEE}, s_{initMEE}, T, A)$, où :

- $V_I$ est l'ensemble non vide des variables d'entrée, (Cardinal de $V_I$ : $|V_I| = n_{V_I}$).
- $V_O$ est l'ensemble non vide des variables de sortie, ($|V_O| = n_{V_O}$).
- $S_{MEE}$ est l'ensemble des étapes, ($|S_{MEE}| = n_{S_{MEE}}$).
- $s_{initMEE} \in S_{MEE}$ est l'étape initiale.
- $T$ est l'ensemble des transitions $t$.
- $A$ est l'ensemble des actions $a$.

Par construction, $V_I$, $V_O$ et $S_{MEE}$ sont des ensembles distincts non vides.

Une transition $t$ de $T$ est définie par le triplet :
$t = (Am(t), Av(t), R(t))$, où :

- $Am(t)$ est l'étape en amont de la transition $t$, ($Am(t) \in S_{MEE}$).

1. ${I_M}^*$ signifie ${I_M}^n, n \in \mathbb{N}^*$.

– $Av(t)$ est l'étape en aval de la transition $t$, $(Av(t) \in S_{MEE})$
– $R(t)$ est la réceptivité associée à la transition $t$. Une réceptivité $R(t)$ est une expression booléenne formée à partir des seules variables d'entrée $V_I$.

Une action $a$ de $A$ est définie par le couple : $a = (s(a), o(a))$, où :
– $s(a)$ est l'étape à laquelle l'action $a$ est associée, $(s(a) \in S_{MEE})$.
– $o(a)$ est la sortie sur laquelle influe $a$, $(o(a) \in V_O)$.

En suivant cette formalisation, la MEE donnée figure 3 peut être définie par le 6-uplet suivant :

$$\begin{cases} V_I = \{a, b, c\} \\ V_O = \{U, V, W\} \\ S_{MEE} = \{s_1, s_2, s_3, s_4\} \\ S_{initMEE} = s_1 \\ T = \{(s_1, s_2, a.\bar{b}), (s_1, s_3, \bar{a}.(b+c)), (s_2, s_1, \bar{a}.\bar{b}), \\ \quad (s_2, s_4, b), (s_3, s_4, a.b), (s_4, s_1, \bar{a}.\bar{b}.\bar{c})\} \\ A = \{(s_2, U), (s_3, V), (s_4, U), (s_4, W)\} \end{cases} \tag{2}$$

Par construction, la MEE est bien structurée, c'est-à-dire :
– Tout franchissement d'une transition correspond à un changement d'étape :

$$\forall t \in T\, [Am(t) \neq Av(t)] \tag{3}$$

– Les transitions ne sont pas redondantes :

$$\forall (t_i, t_j) \in T^2\, [Am(t_i) = Am(t_j) \Rightarrow Av(t_i) \neq Av(t_j)] \tag{4}$$

– Les actions ne sont pas redondantes :

$$\forall (a_i, a_j) \in A^2\, [s(a_i) = s(a_j) \Rightarrow o(a_i) \neq o(a_j)] \tag{5}$$

– Toute étape de la MEE est atteignable depuis l'étape initiale ; il existe une succession de transitions depuis l'étape initiale vers chaque étape de la MEE :

$$\forall s \in \{S_{MEE} - \{s_{initMEE}\}\}\, \exists (t_1..t_n) \in T^n (n \in \mathbb{N}^*) \left[ \begin{cases} Am(t_1) = s_{initMEE} \\ Av(t_n) = s \\ \forall i \in [1, n-1], Av(t_i) = Am(t_{i+1}) \end{cases} \right] \tag{6}$$

– Les transitions en aval d'une même étape sont deux à deux exclusives (structure de graphe d'état) :

$$\begin{array}{l} \forall s \in S_{MEE} \forall (t_i, t_j) \in T^2 \\ [Am(t_i) = Am(t_j) = s \Rightarrow R(t_i) \cdot R(t_j) = 0] \end{array} \tag{7}$$

– La MEE ne contient pas d'évolution fugace :

$$\begin{array}{l} \forall s \in S_{MEE} \forall (t_i, t_j) \in T^2 \\ [Am(t_i) = Av(t_j) = s \Rightarrow R(t_i) \cdot R(t_j) = 0] \end{array} \tag{8}$$

Pour pouvoir tester de manière non invasive cette MEE, il est nécessaire de pouvoir déterminer quelle est l'étape active par la seule observation de ces sorties. Pour cela, à chaque étape doit être associé un ensemble différent d'actions défini comme suit :

$$\forall (s_i, s_j) \in S_{MEE}^2\, [O(s_i) \neq O(s_j)] \tag{9}$$

où $O(s_i)$ est le sous-ensemble de $V_O$ regroupant toutes les sorties émises lorsque l'étape $s_i$ est active.

$$O(s_i) = \{o(a) \mid \exists a \in A\, [s(a) = s_i]\} \tag{10}$$

### B. Construction de la machine de Mealy

La machine de Mealy $M$ : $(I_M, O_M, S_M, s_{initM}, \delta_M, \lambda_M)$, représentant la machine à états équivalente $MEE$ : $(V_I, V_O, S_{MEE}, s_{initMEE}, T, A)$, est définie comme suit :
– $I_M$ : alphabet d'entrée. Cet alphabet contient $2^{n_{V_I}}$ éléments. Chaque élément $i_i$ de cet alphabet représente une combinaison distincte des variables logiques de $V_I$. Un minterme [2] distinct formé à partir des variables de $V_I$ peut donc être associé à chaque élément $i_i$ (cf. Table I pour l'exemple de la figure 3). Par construction, nous avons :

$$\forall (i_i, i_j) \in I_M^2\, [m_I(i_i) \cdot m_I(i_j) = 0] \tag{11}$$

TABLE I
ASSOCIATION ÉVÈNEMENT D'ENTRÉE/MINTERME

| Ev. entrée | $i_0$ | $i_1$ | $i_2$ | $i_3$ | $i_4$ | ... | $i_7$ |
|---|---|---|---|---|---|---|---|
| Minterme | $\bar{a}.\bar{b}.\bar{c}$ | $\bar{a}.\bar{b}.c$ | $\bar{a}.b.\bar{c}$ | $\bar{a}.b.c$ | $a.\bar{b}.\bar{c}$ | ... | $a.b.c$ |

– $O_M$ : alphabet de sortie. Cet alphabet contient $2^{n_{V_O}}$ éléments. Chaque élément $o_i$ de cet alphabet représente une combinaison distincte des variables logiques de $V_O$. Un minterme distinct formé à partir des variables de $V_O$ est associé à chaque élément $o_i$ (cf. Table II). Par construction, nous avons :

$$\forall (o_i, o_j) \in O_M^2\, [m_O(o_i) \cdot m_O(o_j) = 0] \tag{12}$$

TABLE II
ASSOCIATION ÉVÈNEMENT DE SORTIE/MINTERME

| Ev. sortie | $o_0$ | ... | $o_5$ | $o_6$ | $o_7$ |
|---|---|---|---|---|---|
| Minterme | $\bar{U}.\bar{V}.\bar{W}$ | ... | $U.\bar{V}.W$ | $U.V.\bar{W}$ | $U.V.W$ |

– $S_M$ : ensemble des états. A toute étape de la machine à état équivalent $MEE$ est associée un état de la machine de Mealy $M$. Dans un souci de lisibilité, l'état associé à une étape et cette étape seront notés de manière identique. Ainsi, l'ensemble $S_M$ est identique à l'ensemble $S_{MEE}$.

$$S_M \equiv S_{MEE} \tag{13}$$

– $S_{initM}$ : état initial. Cet état est associé à l'étape initiale de $MEE$.

$$S_{initM} \equiv S_{initMEE} \tag{14}$$

– Les fonctions de transition $\delta_M$ et de sortie $\lambda_M$ sont définies à partir des éléments de $V_I$, $V_O$, $S_{MEE}$, $T$, $A$ et $I_M$, $O_M$, $S_M$. La machine $M$ doit être déterministe et complètement spécifiée, c'est-à-dire :

$$\forall (s, i) \in S_M \times I_M \left[ \begin{cases} \exists! \delta_M(s, i) \in S_M \\ \exists! \lambda_M(s, i) \in O_M \end{cases} \right]^3 \tag{15}$$

2. Un minterme est une expression logique de $n$ variables utilisant uniquement l'opérateur logique ET et l'opérateur complément.
3. $\exists!$ : Il existe un unique élément.

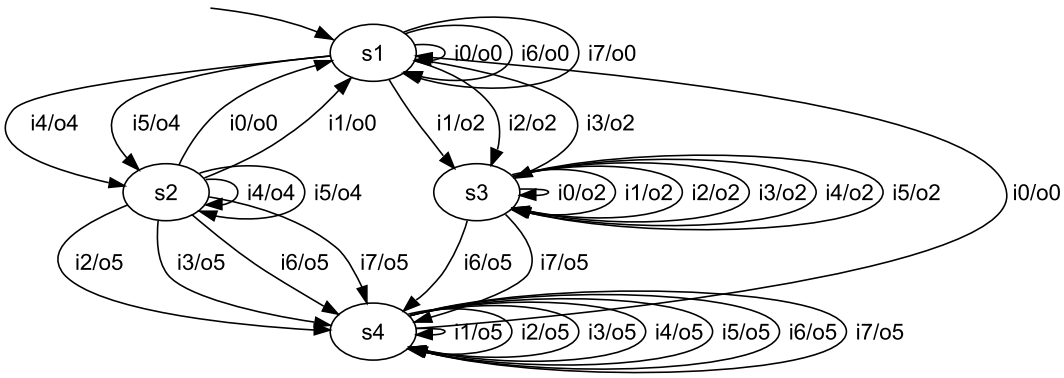



Fig. 5. Représentation graphique de la machine de Mealy correspondant à la MEE présentée Fig. 3

La fonction de transition $\delta_M$ est définie par :

$$\begin{array}{l} \forall s \in S_M \forall i \in I_M \\ \left[ \begin{array}{l} \text{si } \exists t \in T \left[ \left\{ \begin{array}{l} Am(t) = s \\ R(t) \cdot m_I(i) = m_I(i) \end{array} \right. \right] \\ \quad \text{alors } [\delta_M(s,i) = Av(t)] \\ \text{sinon} \\ \quad \text{alors } [\delta_M(s,i) = s] \end{array} \right] \end{array} \tag{16}$$

La fonction de transition $\delta_M$ est complètement spécifiée puisqu'une valeur est définie pour chaque couple $(s,i) \in S_M \times I_M$. Cette valeur est unique car les transitions en aval d'une étape $s$ sont deux à deux exclusives (équation 7).

La construction de la fonction de sortie $\lambda_M$ s'appuie sur le fait que les sorties émises ne dépendent que de l'étape active de $MEE$. La fonction de sortie $\lambda_M$ est définie par :

$$\begin{array}{l} \forall s \in S_M \forall i \in I_M [\lambda_M(s,i) = o_j] \\ \text{où } \forall o_j \in O_M \\ \left[ \left\{ \begin{array}{l} \forall v \in O(\delta_M(s,i)) \, [m_O(o_j) \cdot v = m_O(o_j)] \\ \forall v \in \{V_O - O(\delta_M(s,i))\} \, [m_O(o_j) \cdot \overline{v} = m_O(o_j)] \end{array} \right. \right] \end{array} \tag{17}$$

Les définitions de $\delta_M$ et $\lambda_M$ proposées assurent le déterminisme d'évolution et d'émission des sorties. La machine de Mealy obtenue est donc bien déterministe et complètement spécifiée. De plus, chaque état de la machine de Mealy est atteignable depuis l'état initial puisque cette contrainte est vérifiée pour la MEE (équation 6).

La machine de Mealy correspondant à la MEE de la figure 3 est représentée graphiquement sur la figure 5. Sa description mathématique est définie par le 6-uplet suivant :

$$\left\{ \begin{array}{l} I_M = \{i_0, i_1, i_2, i_3, i_4, i_5, i_6, i_7\} \\ O_M = \{o_0, o_1, o_2, o_3, o_4, o_5, o_6, o_7\} \\ S_M = \{s_1, s_2, s_3, s_4\} \\ S_{initM} = s_1 \\ \delta_M = S_M \times I_M \longrightarrow S_M \\ \lambda_M = S_M \times I_M \longrightarrow O_M \end{array} \right. \tag{18}$$

La définition des fonctions $\delta_M$ et $\lambda_M$ est donnée sous forme tabulaire en table III. A chaque couple $(s,i)$ est associé le couple $(\delta_M(s,i), \lambda_M(s,i))$.

Comme le montre la représentation des fonctions $\delta_M(s,i)$ et $\lambda_M(s,i)$, chaque couple $(s,i) \in S_M \times I_M$ vérifie :

$$\begin{array}{l} \forall ((s,i),(s',i')) \in (S_M \times I_M)^2 \\ [\delta_M(s,i) = \delta_M(s',i') \Rightarrow \lambda_M(s,i) = \lambda_M(s',i')] \end{array} \tag{19}$$

TABLE III
REPRÉSENTATION TABULAIRE DES FONCTIONS $\delta_M$ ET $\lambda_M$

| i \ s | $s_1$ | $s_2$ | $s_3$ | $s_4$ |
|---|---|---|---|---|
| $i_0$ | $s_1$,$o_0$ | $s_1$,$o_0$ | $s_3$,$o_2$ | $s_1$,$o_0$ |
| $i_1$ | $s_3$,$o_2$ | $s_1$,$o_0$ | $s_3$,$o_2$ | $s_4$,$o_5$ |
| $i_2$ | $s_3$,$o_2$ | $s_4$,$o_5$ | $s_3$,$o_2$ | $s_4$,$o_5$ |
| $i_3$ | $s_3$,$o_2$ | $s_4$,$o_5$ | $s_3$,$o_2$ | $s_4$,$o_5$ |
| $i_4$ | $s_2$,$o_4$ | $s_2$,$o_4$ | $s_3$,$o_2$ | $s_4$,$o_5$ |
| $i_5$ | $s_2$,$o_4$ | $s_2$,$o_4$ | $s_3$,$o_2$ | $s_4$,$o_5$ |
| $i_6$ | $s_1$,$o_0$ | $s_4$,$o_5$ | $s_4$,$o_5$ | $s_4$,$o_5$ |
| $i_7$ | $s_1$,$o_0$ | $s_4$,$o_5$ | $s_4$,$o_5$ | $s_4$,$o_5$ |

Ce résultat est la conséquence de deux propriétés de la MEE (les valeurs de sorties ne dépendent que de l'étape active ; à chaque étape est associée un ensemble distinct d'actions) et assure la minimalité de la machine de Mealy obtenue.

La machine de Mealy construite contient autant d'états (ou noeuds de son graphe) qu'il y a d'étapes dans la MEE. Le nombre de transitions (ou arcs de son graphe) ne dépend que du nombre d'étapes et de variables d'entrée de la MEE.

$$\left\{ \begin{array}{l} n_{etats} = n_{S_{MEE}} \\ n_{transitions} = n_{S_{MEE}} \cdot 2^{n_{V_I}} \end{array} \right. \tag{20}$$

Dans l'exemple, la machine de Mealy obtenue contient 4 états et 32 transitions. Il est important de noter que le nombre de transitions de la MEE n'a pas d'influence sur la taille de la machine de Mealy obtenue.

## V. CONSTRUCTION DE LA SÉQUENCE DE TEST

La troisième phase de notre approche est la construction de la séquence de test de la machine de Mealy, obtenue à partir de la MEE, par la méthode du tour de transition ([7]). Une fois cette séquence établie, nous aurons à notre disposition une séquence de sollicitation du SEM à tester qui parcourt la totalité de l'espace d'états de la spécification donnée en Grafcet.

Cette séquence est une solution particulière, pour le graphe correspondant à la structure de la machine de Mealy, d'un problème d'optimisation bien connu en théorie des graphes : le problème du postier chinois ([8] et [9]). Le problème peut être formulé de manière générale comme

suit : Déterminer un circuit fermé traversant chaque arc du graphe au moins une fois.

Le graphe décrivant la structure d'une machine de Mealy est orienté mais non pondéré, ce qui simplifie le problème d'optimisation. Dans ce cas, il a été démontré ([8] et [9]) que la solution optimale est le cycle eulérien[4] de longueur minimale, si le graphe est eulérien (les degrés entrant et sortant de chaque noeud sont égaux). Lorsque le graphe n'est pas eulérien (au moins un noeud a des degrés entrant et sortant différents), il est nécessaire de le rendre eulérien en dupliquant certains arcs. Cette transformation consiste à déterminer la plus courte séquence reliant par paire les noeuds dont les degrés entrant et sortant sont différents et à la dupliquer autant de fois que nécessaire.

Pour le graphe obtenu (cf. Fig. 5), le degré entrant est supérieur au degré sortant pour les états $s_3$ et $s_4$ (la différence vaut respectivement +1 et +5), tandis que le degré entrant est inférieur au degré sortant pour les états $s_1$ et $s_2$ (la différence vaut respectivement -2 et -4). La structure de cette machine de Mealy est donc décrite par un graphe non eulérien. Pour obtenir un graphe eulérien, certains arcs ont été ajoutés comme suit :

1. Le noeud correspondant à l'état $s_4$ est le noeud dont la différence entre les degrés entrant et sortant est la plus grande. La seule solution pour équilibrer ce noeud est de dupliquer 5 fois l'arc étiqueté $i_0/o_0$ de $s_4$ vers $s_1$. Le degré de $s_1$ devient alors +3.

2. Ensuite, le noeud dont la différence entre les degrés entrant et sortant est la plus grande est le noeud correspondant à l'état $s_2$. Une solution pour équilibrer ce noeud est de dupliquer 4 fois l'arc étiqueté $i_5/o_4$ de $s_1$ vers $s_2$. Le degré de $s_1$ devient alors -1.

3. Enfin, les noeuds correspondant aux états $s_1$ et $s_3$ sont équilibrés en dupliquant 1 fois l'arc étiqueté $i_5/o_5$ de $s_3$ vers $s_4$ et 1 fois l'arc étiqueté $i_0/o_0$ de $s_4$ vers $s_1$.

Ainsi, 11 arcs ont été ajoutés. En pratique, dupliquer un arc revient à le traverser plusieurs fois. Les arcs étiquetés $i_0/o_0$ de $s_4$ vers $s_1$, $i_5/o_4$ de $s_1$ vers $s_2$ et $i_5/o_5$ de $s_3$ vers $s_4$ sont donc réutilisés respectivement 6, 4 et 1 fois dans la séquence de test.

Pour l'exemple traité, la séquence de test de longueur minimale construite démarre de l'état $s_1$ et nécessite 43 (32+11) évolutions pour traverser tous les arcs du graphe et ramener la machine à son état initial (cf. Table IV).

A partir de la séquence de test obtenue, il est possible de tester le SEM de manière exhaustive de la manière suivante. Une fois le SEM initialisé, le banc de test le sollicite avec des signaux d'entrée, construits à partir des mintermes associés aux évènements d'entrée donnés dans la deuxième ligne de la table IV. Le banc de test compare ensuite les signaux de sortie délivrés par le SEM testé aux signaux de sortie attendus, construits à partir des mintermes associés aux évènements de sortie donnés dans la troisième ligne de la table IV. L'association d'une combinaison de sorties différente à chaque état interne du SEM permet de détecter à la fois les fautes de sortie et les fautes de transfert. Dans ce cas, la fonction *status* n'est plus nécessaire pour détecter ces dernières.

4. Un cycle eulérien est une séquence ayant même noeud de départ et d'arrivée et traversant une seule fois chaque arc du graphe.

TABLE IV

Séquence de test de la machine de Mealy obtenue

| Pas | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Entrée | $i_4$ | $i_5$ | $i_4$ | $i_1$ | $i_7$ | $i_6$ | $i_0$ | $i_5$ | $i_0$ | $i_1$ |
| Sortie | $o_4$ | $o_4$ | $o_4$ | $o_0$ | $o_0$ | $o_0$ | $o_0$ | $o_4$ | $o_0$ | $o_2$ |

| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i_5$ | $i_4$ | $i_3$ | $i_2$ | $i_1$ | $i_0$ | $i_7$ | $i_7$ | $i_6$ | $i_5$ | $i_4$ | $i_3$ |
| $o_2$ | $o_2$ | $o_2$ | $o_2$ | $o_2$ | $o_2$ | $o_5$ | $o_5$ | $o_5$ | $o_5$ | $o_5$ | $o_5$ |

| 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i_2$ | $i_1$ | $i_0$ | $i_5$ | $i_7$ | $i_0$ | $i_5$ | $i_6$ | $i_0$ | $i_5$ | $i_3$ | $i_0$ |
| $i_5$ | $o_5$ | $o_0$ | $o_4$ | $o_5$ | $o_0$ | $o_4$ | $o_5$ | $o_0$ | $o_4$ | $o_5$ | $o_0$ |

| 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i_5$ | $i_2$ | $i_0$ | $i_3$ | $i_7$ | $i_0$ | $i_2$ | $i_6$ | $i_0$ | | | |
| $o_4$ | $i_5$ | $o_0$ | $o_2$ | $o_5$ | $o_0$ | $o_2$ | $o_5$ | $o_0$ | | | |

## VI. Conclusion

Le principal apport de ce travail est la définition d'une technique de transcription formelle d'un grafcet en une machine de Mealy, permettant ainsi la génération automatique d'une séquence de test exhaustif du comportement d'un SEM spécifié en Grafcet. Nos travaux actuels portent sur trois thèmes complémentaires :

- la mise en oeuvre expérimentale de cette procédure de test dans un environnement réel afin de juger de l'influence de la prise en compte du cycle de scrutation du SEM ;
- la génération de séquences de test dans le cas de machines de Mealy dont l'état courant ne peut être connu uniquement par observation des sorties ;
- l'extension de la classe de Grafcet étudiée pour permettre la prise en compte des spécifications faisant référence au temps.

## Références


[1] Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker et Alexander Pretschner, éditeurs. *Model-Based Testing of Reactive Systems, Advanced Lectures*, volume 3472 de *Lecture Notes in Computer Science*. Springer, 2005.

[2] E. Brinksma, R. Alderden, R. Langerak, J. van de Lagemaat et J. Tretmans : A formal approach to conformance testing. *In J. de Meer, L. Mackert, and W. Effelsberg, editors, Second International Workshop on Protocol Test Systems*, pages 349–363, 1990.

[3] D. Lee et M. Yannakakis : Principles and methods of testing finite state machines - a survey. *In Proceedings of the IEEE*, volume 84, pages 1090–1123, 1996.

[4] Jan Tretmans : Model based testing with labelled transition systems. *In* Robert M. Hierons, Jonathan P. Bowen et Mark Harman, éditeurs : *Formal Methods and Testing*, volume 4949 de *Lecture Notes in Computer Science*, pages 1–38. Springer, 2008.

[5] IEC60848 : *GRAFCET specification language for sequential function charts*. Numéro 2. International Electrotechnical Commission, 2002.

[6] J.M. Roussel : *Analyse De Grafcets Par Génération Logique De l'Automate Equivalent*. Thèse de doctorat, Ecole Normale Supérieure de Cachan, France, 1994. http ://tel.archives-ouvertes.fr/tel-00340842.

[7] S. Naito et M. Tsunoyama : Fault detection for sequential machines by transitions tours. *In Proceedings of the IEEE Fault Tolerant Computer Symposium*, pages 238–243, 1981.

[8] K. Mei-Ko : Graphic programming using odd or even points. *Chinese Mathematics*, 1:273–277, 1962.

[9] J. Edmonds et E. L. Johnson : Matching, Euler tours and the chinese postman. *Mathematical Programming*, 5:88–124, 1973.