

---

## Identification comportementale des systèmes logiques en vue de leur surveillance

Stéphane Klein\*\*\*\* – Jean-Jacques Lesage\* – Lothar Litz\*\*

\* *Laboratoire Universitaire de Recherche en Production Automatisée, ENS de Cachan*  
61 avenue du Président Wilson, F-94235 Cachan Cedex  
{klein, lesage}@hurpa.ens-cachan.fr

\*\* *Institute of Automatic Control, University of Kaiserslautern*  
Erwin-Schrödinger-Str. 12, D-67663 Kaiserslautern  
{sklein, litz}@eit.uni-kl.de

---

*RÉSUMÉ. Cet article présente une méthode d'identification de systèmes logiques composés d'un processus opératif et d'un contrôleur fonctionnant en boucle fermée. L'utilisation du modèle obtenu étant la surveillance en ligne, la maîtrise de la qualité de l'identification est fondamentale. Afin de quantifier la cohérence entre le modèle identifié et le comportement observé sur le système, un indicateur de justesse comportementale est introduit. De plus, la taille du modèle identifié devant rester compatible avec une utilisation réactive en ligne, deux indicateurs de complexité structurelle sont définis pour la quantifier. L'algorithme d'identification proposé permet, grâce à l'utilisation de ces indicateurs, d'ajuster la granularité du modèle identifié. Des résultats expérimentaux sont présentés et discutés.*

*ABSTRACT. This article presents a method to identify logic discrete event systems composed of a controller and a plant running in a closed loop. Since the identified model will be used for fault detection, the quality of the identification is a major issue. In order to quantify the accuracy between the identified model and the observed behaviour of the system, a behavioural quality metric is defined. Furthermore, the size of the identified model must be compatible for an online use. Therefore, two criteria that define the structural complexity of the identified model are introduced. Thanks to these criteria, the proposed identification algorithm allows defining the accuracy of the identified model. Finally experimental results are presented and discussed.*

*MOTS-CLÉS : Identification, Systèmes à événements discrets, Systèmes logiques, Justesse comportementale, Complexité structurelle, Surveillance.*

*KEYWORDS: Identification, Discrete event systems, Logic systems, Behavioural accuracy, Structural complexity, Fault detection.*

---

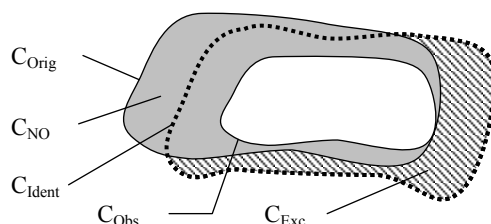
## 1. Introduction

Pour assurer une rentabilité maximale des systèmes de production, il convient d'en éviter les arrêts intempestifs. L'un des moyens couramment utilisés pour y parvenir est d'assurer une surveillance en ligne du système de production afin de détecter ses dérives comportementales et de déterminer ainsi les interventions préventives ou correctives nécessaires à son bon fonctionnement. De nombreuses techniques de surveillance en ligne des processus reposent sur des approches à *base de modèles* (Dubuisson, 2001). Le principe de ces approches est de comparer l'évolution réelle observée d'un système avec son évolution théorique, évaluée à partir d'un modèle de bon comportement de ce système. Une des difficultés rencontrées avec ces approches est l'obtention du modèle de comportement. Deux solutions sont envisageables. La première consiste à construire un *modèle de connaissance*, comme le font pour les Systèmes à Événements Discrets (SED) : (Lunze *et al.*, 2001; Sztipaovits and Misra, 1996; Sampath *et al.*, 1996). Dans ce cas, des modèles de comportement des composants élémentaires du système et de leurs interactions sont construits (le plus souvent sous formes d'automates). Le comportement du système complet est ensuite obtenu par composition des automates de composants. Cette solution est peu applicable aux systèmes de grande taille qui comportent un nombre important de composants élémentaires en interaction. D'autre part, la composition d'automates présente l'inconvénient majeur de générer de nombreux états n'ayant aucun sens physique qui « polluent » le modèle utilisé en surveillance. La seconde solution est de construire un modèle à partir de l'observation du comportement du système réel. Cette démarche, nommée *identification*, ne nécessite pas de connaissance a priori du système dont on souhaite obtenir un modèle de comportement. De plus, ce modèle étant construit à partir de la seule observation du système en fonctionnement, il traduit généralement assez fidèlement le comportement théorique attendu du système.

Ce papier a pour objectif de présenter une technique d'identification du comportement d'une classe de SED : les *systèmes logiques*. Nous supposons ces systèmes logiques composés d'un processus opératif à fonctionnement cyclique et d'un contrôleur interagissant en boucle fermée (cf. figure 2). Nous présentons dans un premier temps les difficultés majeures rencontrées dans le cadre de l'identification de comportement pour les SED. Les fondements théoriques de notre approche sont ensuite définis dans la section 3 et l'algorithme d'identification que nous proposons est présenté dans la section 4. Le problème de la réduction de l'écart entre le comportement observé sur le système réel et le comportement traduit par l'automate identifié est plus particulièrement développé dans la section 5. Pour terminer, des résultats expérimentaux sont détaillés et discutés.

## 2. Problématique de l'identification comportementale des SED

Pour décrire les deux difficultés majeures rencontrées dans une démarche d'identification du comportement d'un SED, nous nous appuyons sur la représentation graphique de la figure 1.



**Figure 1.** *Comportements originels, observés et identifiés d'un SED*

Soient : ( $C_{Orig}$ ) l'ensemble des *comportements originels*, postulés mais non connus, du SED réel,  
 ( $C_{Obs}$ ) l'ensemble des *comportement observés* expérimentalement sur le système réel,  
 ( $C_{Ident}$ ) l'ensemble des *comportements identifiés*, obtenu par application d'un algorithme d'identification à partir de  $C_{Obs}$ .

Le premier problème rencontré est inhérent au non déterminisme des systèmes de production auxquels nous nous intéressons. L'ensemble des états et des trajectoires entre états de tels SED ne peut en effet pas être parcouru en un temps borné. Il faudrait donc en théorie une durée d'observation infinie pour pouvoir construire par identification leur comportement de manière exhaustive (c'est-à-dire construire  $C_{Orig}$ ). En pratique nous ne pourrions pratiquer l'identification qu'à partir d'une observation incomplète du comportement du SED réel. Nous sommes donc conduit à considérer que  $C_{Obs} \subset C_{Orig}$ , ou, en appelant  $C_{NO} = C_{Orig} \setminus C_{Obs}$  l'ensemble des comportements non observés, à considérer que  $C_{NO} \neq \emptyset$  (zone grisée sur la figure 1).

Le second problème rencontré est celui de la différence entre les grandeurs observables sur un SED et le concept d'état qui est le plus souvent retenu pour sa modélisation comportementale. En effet, il est expérimentalement aisé d'observer l'évolution d'un SED au travers de l'évolution des valeurs prises par ses entrées et ses sorties. La transcription du comportement observé, qui peut être représenté comme dans (Lee *et al.*, 2003) par l'ensemble des séquences de ses *vecteurs d'entrées/sorties*, dans un modèle manipulable de type automate ou Réseau de Petri introduit inévitablement des écarts dus aux abstractions et aux adaptations sémantiques qu'il convient de réaliser pour parvenir à la notion d'état. Donc  $C_{Obs} \neq C_{Ident}$ . Pour que  $C_{Ident}$  puisse être utilisé en surveillance, nous devons donc :

- 1) vérifier que  $C_{Obs} \subset C_{Ident}$ . Cette relation, qui traduit la propriété de simulation énoncée par (Lee *et al.*, 2003), indique que le modèle est

*complet* au sens de (Blanke et al., 2003) et qu'il est utilisable à des fins de détection d'erreurs.

- 2) maximiser la *justesse* du modèle en minimisant  $C_{\text{Ident}} \setminus C_{\text{Obs}}$ . En effet,  $C_{\text{Ident}} \setminus C_{\text{Obs}}$  peut contenir des comportements originels non observés (ce qui est favorable pour la surveillance), mais peut également contenir des comportements « excédentaires », que l'on peut définir par :  $C_{\text{Exc}} = C_{\text{Ident}} \setminus (C_{\text{Orig}} \cap C_{\text{Ident}})$  (zone hachurée sur la figure 1). Ces comportements excédentaires n'ont aucun sens par rapport au comportement originel du système surveillé. En cours de surveillance, ils seront cependant observés comme étant cohérents et conduiront à la non-détection d'erreurs.

Dans la suite de cet article, et compte tenu que nous nous intéressons aux systèmes de production à fonctionnement cyclique, nous considérons que nous observons le comportement du système réel sur un nombre de cycles suffisamment grand pour pouvoir faire l'hypothèse que  $C_{\text{NO}} \approx \emptyset$ . Nous développons donc plus particulièrement l'approche que nous avons retenue pour maximiser la justesse du modèle obtenu par identification.

### 3. Concepts et modèles utilisés

#### 3.1. Un bref état de l'art

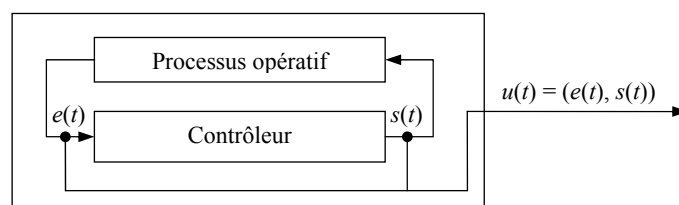
Les recherches bibliographiques que nous avons conduites ne nous ont pas permis de trouver de résultats de recherches ayant à la fois comme objectif la surveillance des SED (ou des systèmes logiques) et l'identification comme approche de modélisation. Outre quelques travaux antérieurs réalisés au LURPA (De Smet *et al.*, 1999), les seuls travaux d'identification relatifs aux systèmes automatisés que nous ayons trouvés sont ceux du CINVESTAV - Mexico. Ainsi, dans (Meda *et al.*, 1998 ; Meda-Campana *et al.*, 2000), les réseaux de Petri sont utilisés pour valider par identification le comportement attendu d'un processus opératif.

Par ailleurs, dans les années 60 et 70 quelques équipes de recherche en informatique ont travaillé dans le domaine de l'identification de langages sous la forme d'automates de Moore ou de Mealy (Kella, 1971; Biermann and Feldman, 1972; Veelenturf, 1978; Booth, 1967). Nous nous sommes bien entendu inspirés de ces approches, mais une particularité importante des systèmes cibles de nos travaux ne nous a pas permis d'en utiliser directement les résultats ; elle va être décrite dans la section suivante.

#### 3.2. Frontières d'isolement et caractéristiques du système identifié

Identifier un langage consiste à construire une relation causale entre les entrées et les sorties d'un système dont on fait l'hypothèse qu'il procure une réaction

immédiate, par un changement de ses sorties, suite à une évolution de ses entrées. Dans ce cas, les automates de Moore ou de Mealy sont parfaitement adaptés. Par contre, dans notre approche nous nous intéressons à l'identification du comportement d'un système automatisé composé d'un contrôleur et d'un processus opératif qui interagissent en boucle fermée (cf. figure 2). La seule observation que nous puissions faire de ce système en fonctionnement est celle de son comportement externe, donné par l'évolution des données échangées entre ces deux sous-systèmes.



**Figure 2.** *Système identifié*

Les observations de l'évolution du système global prennent donc la forme d'une suite ordonnée de signaux d'entrée et de sortie du contrôleur, représentés à chaque instant par un vecteur E/S noté  $u_i(t)$ . Dans cette notation,  $i$  représente le cycle de production courant. Chaque observation  $u_i(t)$  est faite alors que le contrôleur est dans un état stable. La suite ordonnée des vecteurs E/S est nommée une séquence d'observation  $\sigma_i$  telle que  $\sigma_i = (u_i(1), u_i(2), \dots, u_i(|\sigma_i|))$  avec  $|\sigma_i|$  la longueur de la séquence. Il est à noter que les séquences observées sont de longueur variable. Les systèmes considérés fonctionnant de manière cyclique, les cycles de production observés débutent et finissent tous avec la même valeur du vecteur E/S. Nous considérerons qu'à la fin de chaque cycle de production, le système est dans le même état qu'en début de cycle. Ceci se traduit par la relation  $\forall (i, j), u_i(1) = u_j(1) = u_i(|\sigma_i|) = u_j(|\sigma_j|)$ . L'ensemble des observations  $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_{\mathbb{N}}\}$  constitue la base de données expérimentales à partir de laquelle l'identification est réalisée.

Ainsi décrit le contexte expérimental de nos travaux, il apparaît que le système automatisé dont nous cherchons à construire un modèle de comportement doit être considéré comme un *générateur spontané d'événements* (Booth, 1967). Le comportement recherché ne peut donc pas être modélisé à l'aide d'un automate établissant une relation entrée/sortie, comme un automate de Moore ou de Mealy, mais plutôt par un automate autonome permettant de générer une séquence de sortie. De plus, cet automate doit être non-déterministe puis qu'il décrit le comportement d'un système composé d'un contrôleur au comportement déterministe couplé à un processus opératif au comportement non déterministe. Nous avons donc défini une classe d'automates adaptée à notre besoin, nommée *Automate autonome non déterministe à fonction de sortie*, noté par la suite NDAAO pour *Non-Deterministic Autonomous Automaton with Output*.

## 2.2. Les automates autonomes non déterministes à fonction de sortie

Ce modèle, basé sur la classe des automates autonomes non déterministes (Litz, 2004), est défini formellement comme:

$\text{NDAAO} = (\mathbf{X}, \mathbf{\Omega}, f_{\text{nd}}, \lambda, x_0, x_f)$  avec:

$\mathbf{X}$  : ensemble fini des états,

$\mathbf{\Omega}$  : alphabet de sortie,

$f_{\text{nd}}: \mathbf{X} \rightarrow 2^{\mathbf{X}}$  : fonction de transfert,

$\lambda: \mathbf{X} \rightarrow \mathbf{\Omega}$  : fonction de sortie,

$x_0$  : état initial (unique),

$x_f$  : état final (unique).

Dans notre cas, l'état initial  $x_0$  permettra de représenter le premier vecteur d'E/S de chaque séquence observée. L'état final  $x_f$  représentera de même le dernier vecteur de cette séquence. Bien qu'étant associés à la même valeur du vecteur E/S, ces deux états seront distingués dans le processus d'identification.

Les règles d'évolution de cette classe d'automates sont définies de la manière suivante :

Etant dans un état courant  $x_i$ , l'automate peut évoluer dans tout état  $x_j$  tel que  $x_j \in f_{\text{nd}}(x_i)$ . Quand plusieurs états sont atteignables, le choix entre ces états n'est pas déterminé.

La fonction de sortie  $\lambda$  associe chaque état  $x_i$  de l'automate à un élément de l'alphabet de sortie. Cet alphabet est composé de l'ensemble des vecteurs E/S  $u_i(t)$  observés. Comme dans le cas d'une machine de Moore, la sortie est active aussi longtemps que l'automate se trouve dans l'état correspondant.

A l'instar des automates déterministes, le langage accepté par un NDAAO peut être défini de la manière suivante. Notons  $L_{x_i}^n$  l'ensemble des mots de longueur  $n$  que le NDAAO peut accepter en partant de l'état  $x_i$  et  $L^n$  l'ensemble des mots de longueur  $n$  acceptés par le NDAAO. Formellement, ces langages sont définis comme:

$$L_{x_i}^n (\text{NDAAO}) = \left\{ s \in \mathbf{\Omega}^n : s = (\lambda(x(1)), \lambda(x(2)), \dots, \lambda(x(n))) : \left[ \begin{array}{l} \exists (x(1), x(2), \dots, x(n)) : x(1) = x_i \in \mathbf{X}, \\ \forall 1 \leq t \leq n-1, x(t+1) \in f_{\text{nd}}(x(t)) \end{array} \right] \right\},$$

et,

$$L^n = \bigcup_{x_i \in \mathbf{X}} L_{x_i}^n .$$

## 4. Algorithme d'identification

### 4.1. Exemple illustratif

Afin d'illustrer l'utilisation de cet algorithme, nous utiliserons l'exemple d'un système {contrôleur + processus opératif} élémentaire dont le contrôleur manipule deux signaux d'entrée et un signal de sortie (figure 3).

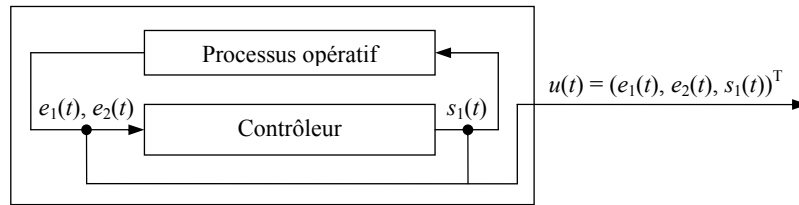


Figure 3. Exemple illustratif

Chaque vecteur E/S observé est codé de la façon suivante :  $u_i(t) = \begin{pmatrix} e_1(t) \\ e_2(t) \\ s_1(t) \end{pmatrix}$ . Deux

séquences observées de vecteurs E/S seront utilisées par la suite. Ces deux séquences sont :

$$\sigma_1 = \left( \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right), \text{ et}$$

$$\sigma_2 = \left( \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right).$$

Afin de simplifier les notations, les vecteurs E/S sont codés avec les lettres A, B, C, D et E. Ces lettres représentent l'alphabet observé. Avec cette notation, les séquences observées sont :  $\sigma_1 = (A, B, C, D, E, A)$  et  $\sigma_2 = (A, D, B, C, D, A, C, A)$ .

### 4.2. Algorithme d'identification

Dans cette partie, notre algorithme d'identification est présenté. Afin d'évaluer et d'optimiser la *justesse* du modèle identifié (au sens défini dans la section 2 de cet article), nous utilisons un paramètre  $k$  caractérisant la longueur des mots reconnus par l'automate. Cette approche est inspirée de celle de (Booth, 1967) qui utilise un

tel paramètre pour l'identification de langages représentés par une machine de Mealy.

Notre algorithme est composé de 5 étapes principales:

1. Pour chaque séquence observée, transformation de la séquence de vecteurs E/S en une séquence de séquences de mots de longueur  $k$  (choisie par l'utilisateur).
2. Traduction de chaque séquence de mots par un automate (NDAAO).
3. Réécriture de la fonction de sortie.
4. Redéfinition de l'état final.
5. Réduction par fusion des états équivalents.

Nous allons maintenant détailler chaque étape de l'algorithme d'identification en l'appliquant sur notre exemple.

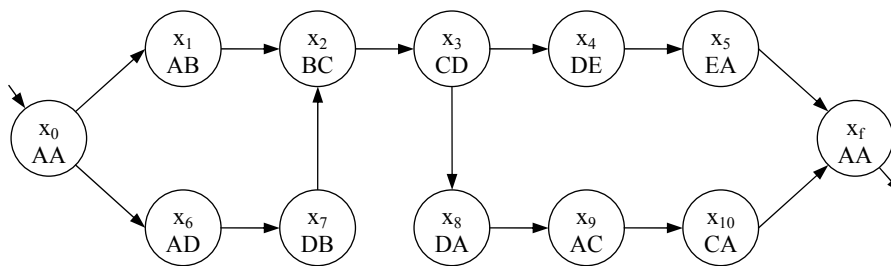
Etape 1 : elle consiste à transformer les séquences  $\sigma_i$  de vecteurs E/S observées en séquences de mots de longueur  $k$ , notées  $\sigma_i^k$ . Puisque toutes les séquences observées commencent et se terminent avec le même mot, le premier et le dernier vecteur E/S de chaque séquence sont dupliqués  $k-1$  fois.

Dans le cas de l'exemple présenté précédemment et pour  $k=2$ , les séquences de mots de longueur  $k$  obtenues sont :

$$\sigma_1^2 = ((A, A), (A, B), (B, C), (C, D), (D, E), (E, A), (A, A)), \text{ et}$$

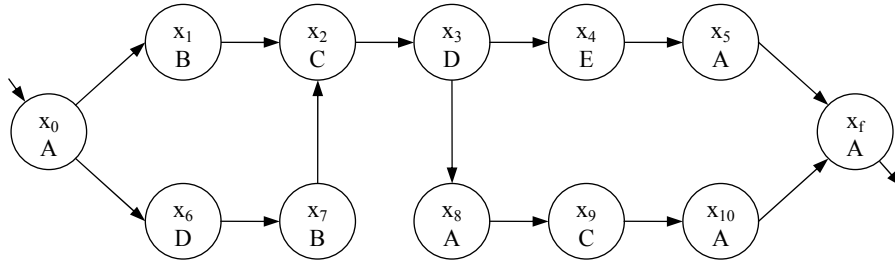
$$\sigma_2^1 = ((A, A), (A, D), (D, B), (B, C), (C, D), (D, A), (A, C), (C, A), (A, A))$$

Etape 2 : un NDAAO qui associe chaque mot de la séquence de mots  $\sigma_i^k$  définie à l'étape 1 à un état de l'automate est construit (cf. figure 4). Le premier mot de chaque séquence est associé à  $x_0$  et le dernier à  $x_f$ .



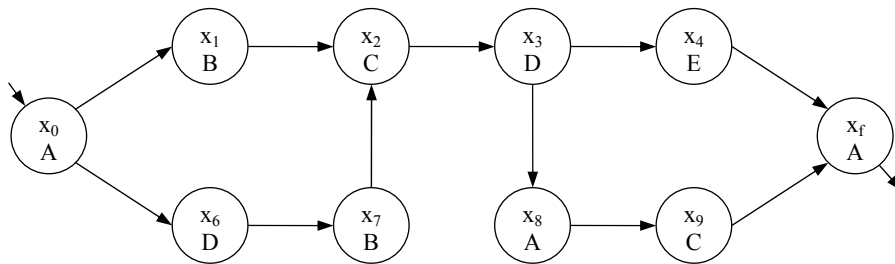
**Figure 4.** Automate identifié initial

Etape 3 : chaque état du NDAAO construit à l'étape 2 correspond à une valeur unique et stable du vecteur d'E/S du système, donnée par la dernière lettre de chaque mot de longueur  $k$ . Aussi, la sortie de l'état correspondant est renommée avec cette valeur.



**Figure 5.** Automate identifié après réécriture de la fonction de sortie

Etape 4 : dans le NDAAO construit à l'étape précédente, les  $k$  états de chaque branche précédant  $x_f$  sont associés à la même valeur du vecteur E/S. Ces états, utilisés comme un artifice de construction peuvent à présent être réduits. Pour cela, la procédure suivante doit être répétée  $k - 1$  fois. Premièrement l'ensemble des états précédents l'état final  $x_f$  est fusionné en un état unique nommé  $x_{f-1}$ . Deuxièmement, l'état  $x_f$  est supprimé de l'ensemble  $\mathbf{X}$  des états. Enfin, l'état  $x_{f-1}$  devient le nouvel état final et est renommé  $x_f$ . Pour l'automate identifié, les états  $x_5$  et  $x_{10}$  sont fusionnés pour former le nouvel état final  $x_f$ .



**Figure 6.** Automate identifié après redéfinition de l'état final

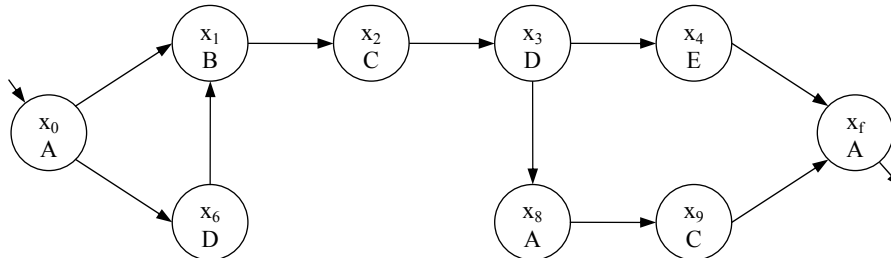
Etape 5 : lors de l'étape 3, plusieurs états peuvent être associés à une même valeur du vecteur E/S. Si ces états sont équivalents, il est judicieux de les fusionner afin de réduire le nombre d'états. Deux états  $x_i$  et  $x_j$  sont équivalents si et seulement si :

1. la valeur de leur fonction de sortie est identique, i.e.  $\lambda(x_i) = \lambda(x_j)$ .
2. leurs ensembles d'états successeurs sont identiques, i.e.  $f_{nd}(x_i) = f_{nd}(x_j)$ .

Il a été prouvé que la fusion d'états remplissant ces deux conditions n'affecte pas les langages acceptés par le NDAAO (Klein, 2005).

Dans l'exemple considéré, les états candidats à la fusion sont les ensembles d'états  $\{x_0, x_8, x_f\}$ ,  $\{x_1, x_7\}$ ,  $\{x_2, x_9\}$  et  $\{x_3, x_6\}$ . Ces quatre ensembles d'états remplissent la première condition, à savoir qu'ils sont associés à la même valeur du

vecteur E/S. Seul le couple d'états  $\{x_1, x_7\}$  remplit également la seconde condition. Seuls ces deux états peuvent donc être fusionnés.



**Figure 7.** Automate identifié après fusion des états équivalents

## 5. Réduction de l'écart entre le comportement observé et l'automate identifié

Nous avons présenté dans la section 2 les problèmes essentiels rencontrés en identification des SED. La difficulté de faire coïncider le comportement observé sur le système réel et le comportement traduit par le modèle identifié est particulièrement importante. L'écart entre ces deux comportements, qui devra être minimisé, peut être caractérisé par la *justesse* du modèle identifié. Nous allons maintenant définir comment évaluer cette justesse.

Deux types d'écarts de justesse peuvent être rencontrés (cf. section 2), caractérisés d'une part par la non-inclusion du comportement observé dans le comportement identifié et d'autre part pas la présence de comportements excédentaires, c'est-à-dire de comportements traduits dans le modèle d'identification et non observés. L'inclusion du comportement observé dans le comportement identifié pouvant être démontrée (Klein, 2005), la maîtrise de cet écart de justesse potentiel ne pose pas de problème particulier. Nous allons plus particulièrement examiner le problème des comportements excédentaires.

Ces comportements peuvent être de trois types :

- il peut s'agir de bons comportements du système qui n'ont pas été observés ; ils doivent être conservés dans le modèle d'identification,
- il peut s'agir de comportements n'ayant aucun sens physique pour le système en question qui doivent être éliminés du modèle d'identification,
- il peut enfin s'agir de comportements fautifs du système. Ces comportements doivent être éliminés, ou à défaut minimisés.

Dans le cadre de la surveillance, concernant ces comportements excédentaires nous retiendrons l'hypothèse la plus sécuritaire en considérant que tout comportement non supporté par le modèle identifié est déclaré fautif. Il est donc important, pour des raisons évidentes d'efficacité, de réduire le nombre de

comportements excédentaires. Afin de quantifier ces comportements excédentaires, et en nous inspirant de (Geffroy *et al.*, 1995), nous avons défini un critère de *justesse comportementale*.

Par ailleurs, il est important de limiter la taille de l'automate identifié afin de préserver l'efficacité de la surveillance qui se pratique en ligne. Comme il n'y a naturellement pas unicité de l'automate capable de traduire un comportement donné, un critère de *complexité structurelle* permettant d'évaluer la taille de l'automate est défini afin de choisir parmi les automates solution.

### 5.1. Justesse comportementale

Un moyen d'évaluer la justesse du modèle identifié est de quantifier les comportements de longueur  $n$  acceptés par le modèle qui n'ont pas été observés. La définition du critère de complexité comportementale nécessite donc en préalable une définition formelle des différents comportements introduits dans la figure 1.

#### 5.1.1. Définition formelle des comportements

##### Comportements identifiés

L'ensemble des comportements identifiés de longueur  $n$  peut être caractérisé par l'ensemble des mots de longueur inférieure ou égale à  $n$  acceptés par l'automate identifié. Ainsi, l'ensemble des comportements identifiés de longueur  $n$  est défini comme:

$$C_{\text{ident}}^n = \bigcup_{k=1}^n L^k (\text{NDAAO}).$$

##### Comportements observés

L'ensemble des comportements observés de longueur  $n$  peut être caractérisé par l'ensemble des mots de longueur inférieure ou égale à  $n$  observés sur le système. Si nous définissons le langage observé  $L_{\text{Obs}}^k$  comme l'ensemble des mots de longueur  $k$  observés,

$$L_{\text{Obs}}^k = \bigcup_{\sigma_i \in \Sigma} \left( \bigcup_{j=1}^{|\sigma_i| - k + 1} (u_i(j), u_i(j+1), \dots, u_i(j+k-1)) \right),$$

nous pouvons définir les comportements observés de longueur  $n$  comme:

$$C_{\text{Obs}}^n = \bigcup_{k=1}^n L_{\text{Obs}}^k.$$

### 5.1.2. Définition du critère de justesse comportementale

En termes de comportements, un modèle est *juste* si  $C_{\text{Obs}}^n = C_{\text{Ident}}^n$  est vérifié pour une valeur fixée de  $n$ . Ceci équivaut à  $\forall i \leq n, L_{\text{Obs}}^i = L^i(\text{NDAAO})$ . Ainsi la complexité comportementale au rang  $n$  est définie comme le nombre de mots de longueur  $n$  acceptés par le modèle identifié ramené au nombre de mots de longueur  $n$  observés. Formellement, ce critère est donc défini par :

$$C_B^n = \frac{|L^n(\text{NDAAO})|}{|L_{\text{Obs}}^n|}.$$

Afin d'illustrer le sens pratique de ce critère, supposons que  $C_B^3 = 1,25$ . Cela signifie que le modèle identifié accepte 25 % de mots de longueur 3 de plus que ceux qui ont été réellement observés. En faisant l'hypothèse que ces mots excédentaires correspondent tous à des comportements fautifs (hypothèse sécuritaire énoncée dans la section 5), cela signifie qu'à partir de chacun des états de l'automate la probabilité qu'une erreur non détectable soit survenue deux évolutions plus tard est de 20 % ( $1 - 1/1,25 = 0,20$ ).

### 5.2. Complexité structurelle

La complexité structurelle d'un automate peut être caractérisée par son nombre d'états et sa connexité. Nous définissons donc les deux indicateurs de complexité structurelle suivants :

$C_{S1} = |\mathbf{X}|$  qui représente le nombre d'états du modèle identifié,

$$C_{S2} = \frac{\sum_{x_i \in \mathbf{X}} |f_{\text{nd}}(x_i)|}{|\mathbf{X}|} \text{ qui représente le nombre moyen de transitions par état.}$$

Ces deux critères sont comparables à ceux définis dans (Gilb, 1977). Dans ces travaux, le nombre de modules et le nombre de connections par module ont été définis pour permettre l'évaluation de la complexité structurelle des programmes informatiques.

### 5.3. Propriétés importantes

Nous avons démontré que l'automate identifié, résultat de l'application de l'algorithme présenté en section 4 pour une valeur de  $k$  donnée, supporte exactement le langage observé  $L_{\text{Obs}}^{k+1}$ . Cela signifie que pour une justesse comportementale visée, par exemple  $C_B^4 = 1$ , l'algorithme pourra avantageusement être appliqué avec un  $k$  connu (dans notre exemple  $k = 3$ ).

Nous avons également prouvé que la complexité comportementale du modèle identifié diminue lorsque sa connexité diminue. En effet, la taille du langage de longueur  $n$  généré par l'automate identifié est directement liée à la connexité de ce dernier. Ainsi une amélioration de la justesse comportementale passe nécessairement par une diminution de la connexité du modèle, ce qui implique généralement un accroissement du nombre d'états.

#### 5.4. Applications

Le premier exemple traité est l'une des stations d'une ligne d'assemblage didactisée installée au LURPA de l'ENS de Cachan. Cette station est composée d'un processus opératif de distribution des pièces à traiter par le reste de la ligne, contrôlé par un Automate Programmable Industriel (API) possédant 35 Entrées/Sorties. 18 cycles de production ont été observés.

Plusieurs applications de l'algorithme d'identification ont été réalisées pour différentes valeurs du paramètre  $k$ . La taille des automates résultat ne nous permet pas de les présenter graphiquement, mais les valeurs des différents critères de qualité définis en section 5 sont indiquées dans le tableau 1. La justesse comportementale est calculée, pour chaque valeur de  $k$ , pour des mots reconnus de longueur 1 à 6.

$k$	Justesse comportementale						Complexité structurelle	
	$C_B^1$	$C_B^2$	$C_B^3$	$C_B^4$	$C_B^5$	$C_B^6$	$C_{S1}$	$C_{S2}$
1	1	1	1,05	1,14	1,28	1,46	104	1,23
2	1	1	1	1,05	1,12	1,21	108	1,20
3	1	1	1	1	1,03	1,09	115	1,18
4	1	1	1	1	1	1,03	124	1,16
5	1	1	1	1	1	1	128	1,14

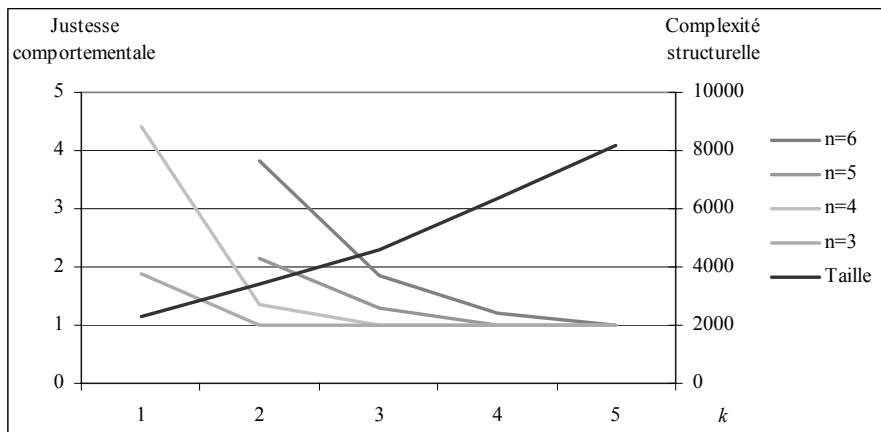
**Tableau 1.** Résultats d'identification pour la ligne d'assemblage du LURPA

L'examen des résultats rassemblés dans le tableau 1 permet de confirmer les propriétés énoncées en section 5.3. Nous avons mis en évidence que le choix du paramètre d'identification  $k$  relevait d'un compromis entre la taille et la justesse du modèle. Dans cet exemple, étant donnée la faible taille des modèles identifiés, il est judicieux de choisir une valeur élevée de  $k$ , par exemple  $k=5$ , de manière à privilégier la justesse de l'identification.

Le second exemple est relatif à une installation industrielle (usine du groupe Freudenberg implanté à Kaiserslautern – Allemagne). Il s'agit d'une station de bobinage d'un matériau non-tissé. L'API qui contrôle la machine a 336 Entrées/Sorties. Un cycle de production génère environ 1000 événements qui confèrent 350 valeurs différentes au vecteur E/S. La durée d'un cycle de production varie de 45 à 120 minutes, selon la longueur de matériau à bobiner. 130 cycles de production ont été observés. Comme pour l'exemple précédent, le tableau 2 présente les résultats obtenus. La justesse comportementale est calculée, pour chaque valeur de  $k$ . La mention « n.c. » signifie « non calculé » et indique que la valeur correspondante n'a pas pu être obtenue en un temps de calcul raisonnable.

$k$	Justesse comportementale						Complexité structurelle	
	$C_B^1$	$C_B^2$	$C_B^3$	$C_B^4$	$C_B^5$	$C_B^6$	$C_{S1}$	$C_{S2}$
1	1	1	1,87	4,40	n.c.	n.c.	2266	1,80
2	1	1	1	1,36	2,15	3,82	3390	1,65
3	1	1	1	1	1,29	1,85	4591	1,52
4	1	1	1	1	1	1,21	6347	1,39
5	1	1	1	1	1	1	8172	1,30

**Tableau 2.** Résultats d'identification pour la ligne de fabrication Freudenberg



**Figure 8.** Evolution comparée de la complexité structurelle et de la justesse comportementale

La figure 8 donne une représentation graphique de l'évolution, en fonction de  $k$ , de la complexité structurelle (donnée sous la forme du nombre d'états de l'automate) et de la justesse comportementale, calculée pour des mots reconnus de longueur  $n$  variant de 3 à 6. Il est clair que l'amélioration de la justesse comportementale s'accompagne d'une augmentation de la complexité structurelle. Cette figure donne une image des compromis à rechercher. Ainsi, si on souhaite maintenir une bonne justesse comportementale pour des mots reconnus de longueur 6 ( $n = 6$ ), tout en souhaitant obtenir un automate relativement compact pour la surveillance, on peut viser l'intersection entre la courbe  $n = 6$  et la courbe de taille de l'automate. Cette intersection est obtenue pour  $2,5 \leq k \leq 2,8$  ; un compromis peut donc être trouvé pour  $k = 3$ .

## 6. Conclusion

Dans un premier temps, nous avons mis en évidence les deux difficultés majeures de l'identification comportementale des SED que sont une observation forcément partielle du comportement réel et l'inclusion, dans le modèle d'identification, de comportements non observés. Nous avons ensuite proposé un algorithme d'identification ainsi que la définition d'indicateurs permettant d'évaluer la qualité du modèle obtenu. L'indicateur de justesse comportementale permet d'évaluer la cohérence entre modèle et réel en calculant le nombre de comportements non observés sur le système et générés par le modèle identifié. Deux critères de complexité structurelle permettent quant à eux d'évaluer la taille de l'automate identifié de manière à s'assurer que sa taille est compatible avec son utilisation en surveillance en ligne du système. La démarche proposée a été illustrée et discutée à l'aide de deux exemples.

Les travaux à venir concernent d'une part la maîtrise de l'observation partielle du système réel et d'autre part la maîtrise de la justesse du modèle. Sur le premier aspect, nous envisageons de tirer parti du fait que durant la surveillance le système est en fonctionnement et que nous pouvons continuer d'observer son comportement. Le développement d'une méthode permettant de compléter le résultat initial de l'identification en intégrant les comportements corrects non encore observés permettra de nous rapprocher de l'hypothèse de durée d'observation infinie. Concernant la justesse du modèle identifié, nos travaux vont porter sur le développement de techniques d'optimisation du paramètre  $k$ .

## 7. Références bibliographiques

- Biermann A.W., Feldman J.A., « On the Synthesis of Finite-State Machines from Samples of their Behavior », *IEEE Transactions on Computers*, vol. 21, 1972, p. 592-597.
- Blanke M., Klinnaert M., Lunze J., Staroswiecki M., *Diagnosis and Fault-Tolerant Control*, Berlin, Heidelberg, New York, Springer-Verlag, 2003.

- Booth T.L., *Sequential Machines and Automata Theory*, New York, London, Sydney, John Wiley and Sons, Inc., 1967.
- Cassandras C., Lafortune S., *Introduction to Discrete Event Systems*, Boston, Dordrecht, London, Kluwer Academic Publishers, 1999.
- Davis R., Hamscher W., *Model-based Reasoning : Troubleshooting , Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, 1988, p. 297-346.
- De Smet O., Roussel J.-M., Hevin, N., « Identification de machine séquentielle binaire : application à un système réactif », *Actes du 2<sup>e</sup> congrès sur la modélisation des systèmes réactifs MSR '99*, Cachan, mars 1999, Paris, Editions Hermès, p. 351-360.
- Dubuisson B., *Diagnostic, intelligence artificielle et reconnaissance de formes*, Paris, Hermès Science Publications, 2001.
- Geffroy J.-C., Baron C., El Maadani K., « Identification de systèmes séquentiels – Une approche unifiée », *Technique et Science Informatiques*, vol. 14, n° 7, 1995, p. 809-837.
- Gilb T., *Software Metrics*, Cambridge, Winthrop Publishers, Inc., 1977.
- Kella J., « Sequential Machine Identification », *IEEE Transactions on Computers*, vol. 20, 1971, p. 332-338.
- Klein S., Identification of Discrete Event Systems for Fault Detection Purposes, Thèse de doctorat, Ecole Normale Supérieure de Cachan – Université de Kaiserslautern (Allemagne), 2005, à paraître.
- Lee E.A., Varaya P., *Structure and Interpretation of Signals and Systems*, Boston, Addison Wesley, 2003.
- Litz L., *Grundlagen der Automatisierungstechnik*, München, Wien, Oldenbourg Verlag, 2004.
- Lunze J., Schröder J., Supavatanakul P., « Diagnosis of Discrete Event Systems : the Method and an Example », *Actes du 12<sup>e</sup> International Workshop on Principles of Diagnosis DX'01*, Via Lattea (Italie), 2001.
- Meda M.-E., Ramirez, A., « Identification in discrete event systems », *Actes de IEEE International Conference on Systems, Man and Cybernetics SMC '98*, San Diego (USA), Octobre 1998, p. 740-745.
- Meda-Campana M.-E., Ramirez-Trevino A., Lopez-Mellado E., « Asymptotic identification of discrete event systems », *Actes du 39<sup>e</sup> IEEE Conference on Decision and Control CDC'00*, Sidney (Australie), 12-15 décembre 2000, papier n°. 1544.
- Sampath M., Sengutpa R., Lafortune S., Sinnamohideen K., Teneketzis, D., « Failure Diagnosis using Discrete-Event Models », *IEEE Transactions on Control Systems Technology*, vol. 4, n° 2, 1996, p. 105-124.
- Veelenturf L.P.J., « Inference of Sequential Machines from Sample Computations », *IEEE Transactions on Computers*, vol. 27, 1978, p. 167-170.