# COnfECt : Une Méthode Pour Inférer Les Modèles De Composants D'un Système

Elliott Blot, Patrice Laurencot, Sébastien Salva

# COnfECt : Une Méthode Pour Inférer Les Modèles De Composants D'un Système.

Elliott Blot, Patrice Laurençot, Sébastien Salva
LIMOS, Université Clermont Auvergne, France
Email: eblot@isima.fr, laurencot@isima.fr,sebastien.salva@uca.fr

**Résumé**

Cet article présente la méthode COnfECt qui complète les méthodes passives d'inférence de modèle pour extraire et représenter les comportements de composants d'un système vu comme une boite noire. COnfECt utilise pour cela les notions d'analyse de traces, de corrélation et de similarité de modèles. Nous montrons également comment intégrer COnfECt à la méthode GK-tail.
**Mots clés :** Inférence de modèle ; Inférence passive ; Composants

## 1 Introduction

L'inférence de modèles formels, aussi appelé *model learning*, regroupe un ensemble de méthodes permettant de retrouver un modèle comportemental d'un système, soit en interagissant avec ce dernier (méthodes actives, e.g., [1]), soit en analysant un ensemble de traces d'exécution obtenu en monitorant le système (méthode passive, e.g., [3]). Un modèle inféré peut ensuite être employé pour analyser le fonctionnement du système ou pour générer des tests. Bien qu'il soit aujourd'hui possible d'inférer des modèles à partir de certains systèmes réels, plusieurs points restent à étudier avant de pouvoir passer dans une phase industrielle. Parmi ceux-ci, nous avons relevé que les méthodes proposées considèrent un système comme une boite noire globale, qui prend des événements d'entrées depuis un environnement et produisent des événements de sorties. Pourtant, la grande majorité des systèmes actuels sont constitués de composants ré-utilisables, qui interagissent entre eux. La représentation de ces composants et de leurs compositions à travers plusieurs modèles permettrait une plus grande lisibilité du fonctionnement du système, voire une plus grande précision.

A travers cet article, nous nous penchons sur cette problématique et introduisons la méthode COnfECt (COrrelate Extract Compose) qui vise à compléter les méthodes d'inférence passives pour produire des systèmes de CEFSMs (Callable Extended FSM). Une CEFSM est une EFSM spécialisée équipée d'un événement interne spécial exprimant l'appel d'une autre CEFSM. L'idée fondamentale utilisée par COnfECt est qu'un composant dans un système peut être identifié par son

comportement. D'une manière simplifiée, COnfECt analyse les traces, détecte les blocs de comportements, extrait ces comportements dans des nouveaux ensembles de traces et aide à produire un ensemble de CEFSMs. COnfECt utilise pour cela les notions d'exploration d'événements (event mining), de corrélation d'événements et de clustering d'ensembles de traces.

Dans cet article, nous présentons sommairement les deux étapes de COnfECt appelées *Analyse de Trace et Extraction* et *Composition de CEFSMs*. Nous montrons enfin un exemple d'intégration de COnfECt avec la méthode GK-tail [3].

La suite de cet article est organisée de la manière suivante : la Section 2 présente les modèles générés par notre méthode, les CEFSMs. La description de l'approche est donnée dans la Section 3. Nous montrons un exemple d'intégration avec la méthode GK-tail, dans la Section 4 et nous concluons et donnons quelques perspectives en Section 5.

## 2 Callable Extended Finite State Machine

Une CEFSM permet de représenter l'appel d'un composant à partir d'un autre composant. Par manque de place, seul un résumé de la définition d'une CEFSM est donnée ici.

Une CEFSM est un tuple $\langle S, S_0, \Sigma, P, T \rangle$ avec $S$ l'ensemble des états, $S_0$ l'état initial, $\Sigma = \Sigma_I \cup \Sigma_O \cup \{call\}$ l'ensemble des symboles avec $call$ un symbole spécial, et $P$ l'ensemble des paramètres. $T$ est l'ensemble des transitions de type $s_1 \xrightarrow{e(p),G} s_2$ avec $e(p)$ un événement composé de paramètres $p \subseteq P$ et $G$ une garde sur $P$ qui doit être satisfaite pour autoriser le franchissement de la transition. Comme dit précédemment, une transition d'une CEFSM $C_1$ peut être annotée par un événement interne dénoté *call(CEFSM)* exprimant le début d'exécution d'une autre CEFSM. Cet événement non observable signifie que la CEFSM appelante $C_1$ est mise en pause tandis que la CEFSM appelée $C_2$ démarre son exécution à partir de son état initial. Quand $C_2$ a atteint un état final, $C_1$ reprend son exécution après l'événement *call(CEFSM)*. Nous ne considérons pas qu'un composant retourne des résultats à un autre composant. Un système de CEFSMs est supposé inclure au moins une CEFSM qui appelle d'autres CEFSMs.

Nous disons également qu'une CEFSM $C$ est callable-complete sur un système de CEFSMs $SC$, si toutes les CEFSMs de $SC$ peuvent être appelées à partir de tous les états de $C$.

## 3 Approche

COnfECt est une approche permettant de générer un système de CEFSMs à partir d'un système en boite noire, afin de compléter les méthodes d'inférence passive. Pour cela, COnfECt analyse les traces obtenues à partir du système à inférer et essaie de détecter les différents composants pour pouvoir les modéliser avec des

CEFSMs. L'outil se base sur les traces obtenues via le système à inférer pour en extraire le comportement des composants. Par conséquent, plus le nombre de traces à disposition est importante, plus l'analyse de traces et l'inférence du modèle sera correcte.

Nous supposons que l'ensemble de traces est collecté de façon synchrone (environnement synchrone). Celles-ci peuvent être obtenues par des outils de monitoring. Dans ce papier, nous ne considérons pas le formatage des traces, et nous considérons que nous avons un "mapper" qui nous fournit des traces comme une séquence d'événements sous la forme $e(p_1 := d_1, ..., p_k := d_k)$ avec $p_1 := d_1, ..., p_k := d_k$ des affectations de paramètres.

COnfECt est composé de deux étapes principales appelées *Analyse de Traces et Extraction*, et *Composition de CEFSMs*, que nous allons décrire ci-dessous.

### 3.1 Analyse de Traces et Extraction

Après l'étude de plusieurs systèmes à base de composants, nous avons généralement observé qu'un composant produit un comportement souvent identifiable du reste des événements dans les différentes traces. De plus, certains composants, notamment dans des systèmes embarqués, produisent des événements non contrôlable (sortie non précédées par des entrées). A partir de ces observations, cette étape a pour but d'identifier les composants dans les traces à l'aide d'un *Coefficient de corrélation*, grâce à la détection des problèmes de contrôlabilité.

Le *coefficient de corrélation* nous permet d'évaluer le lien, ou relation entre deux événements successifs dans des traces. Nous définissons le *coefficient de corrélation* entre deux événements par une fonction, dépendant de facteurs, qui sont définis par les préférences ou les besoins de l'utilisateur. Un exemple de facteur est la fréquence d'apparition d'événements successifs dans les traces, ou encore la similitude entre les paramètres des événements. Ce coefficient est une valeur comprise entre 0 et 1, qui nous permet de déterminer les séquences d'événements dans une trace ayant une forte corrélation s'il est supérieur a un seuil $X$, ou s'ils ont une *faible corrélation* s'il est inférieur a un seuil $Y$. Les seuils $X$ et $Y$ dépendent du contexte, et doivent être déterminés par un expert.

L'étape *Analyse de Trace & Extraction*, dont l'algorithme est donné dans [2], essaie de segmenter chaque trace en séquences d'événements et de reconnaître les séquences d'événements produites par différents composants. Ces dernières séquences sont extraites et placées dans des nouveaux ensembles de traces. Chaque ensemble sera ensuite transformé en CEFSM.

De façon résumée, une trace est segmentée en séquences $\sigma_1 \dots \sigma_k$ de telle sorte qu'une séquence est composé d'événements ayant une forte corrélation et que 2 séquences successives ont une faible corrélation. Nous considérons que ces séquences correspondent à l'appel de composants ayant des comportements différents. L'étape *Analyse de Trace & Extraction* détecte l'appel de ces composants en analysant la corrélation des séquences $\sigma_1 \dots \sigma_k$. Lorsqu'un appel de composant est détecté avec la séquence $\sigma_i \dots \sigma_j$, cette dernière est remplacée par l'événement

$call(CEFSM := C)$ avec $C$ un nouveau composant. La séquence $\sigma_i \dots \sigma_j$ est considérée comme une nouvelle trace qui est placée dans un nouvel ensemble de traces. Cette nouvelle trace est elle même récursivement analysée pour détecter l'appel d'autres composants.

Prenons l'exemple de trace $\sigma$ donné en Figure 1 pour illustrer le fonctionnement de l'étape d'extraction de traces effectuée par la procédure Extraire. Celle-ci prend une trace déjà segmentée et un ensemble de traces $T$ où sera stockée la séquence d'événements résultat. Au départ, nous appelons Extraire($\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6, T$).

**A)** L'algorithme commence avec $\sigma_1$. Supposons que la première séquence qui ait une forte corrélation avec $\sigma_1$ est $\sigma_5$. $\sigma$ est transformé et devient $\sigma_1 call(CEFSM := C_2)\sigma_5\sigma_6$. $Extraire(\sigma_2\sigma_3\sigma_4, T_2)$ est récursivement appelé pour analyser $\sigma' = \sigma_2\sigma_3\sigma_4$.

**B)** $\sigma_2\sigma_4$ ayant une forte corrélation, $\sigma'$ est modifié et devient $\sigma' = \sigma_2 call(CEFSM := C_3)\sigma_4$. La séquence $\sigma_3$ est une nouvelle trace du nouvel ensemble $T_3$. Maintenant que $\sigma'$ est totalement parcourue, elle est ajoutée dans un ensemble de traces $T_2$.

**C)** Nous revenons à la trace $\sigma$, au niveau de la séquence $\sigma_5$ et réappliquons le même procédé. Comme il n'y a plus de séquence qui est fortement corrélée avec $\sigma_5$, la fin de la trace, correspond à un appel de composant. $\sigma_6$, est extraite et placée dans un nouvel ensemble de traces $T_4$. La trace $\sigma$ devient $\sigma = \sigma_1 call(CEFSM := C_2)\sigma_5 call(CEFSM := C_4)$. Elle est placée dans l'ensemble de traces $T_1$.

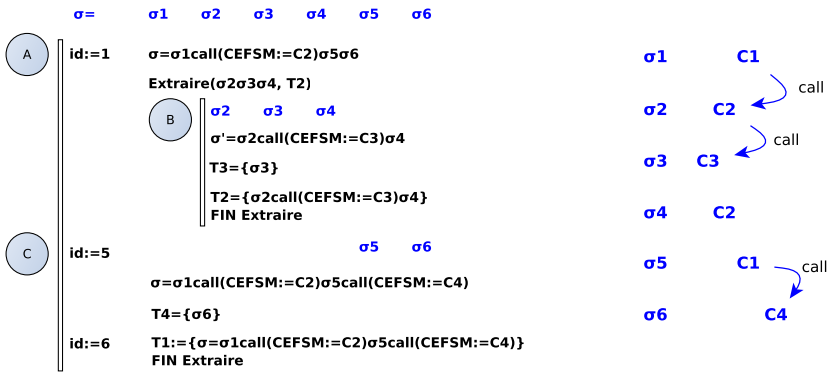


FIGURE 1 – Exemple d'execution de Extraire($\sigma, T$).

A chaque fois que nous avons trouvé des comportements différents dans la trace, nous créons un nouvel ensemble de traces qui seront transformé en CEFSM.

## 3.2 Composition de CEFSMs

Le but de cette étape est d'analyser les compositions de CEFSMs obtenues. Nous proposons trois stratégies de composition dans ce papier.

**Composition stricte :** dans cette stratégie, nous cherchons à ne pas sur-généraliser le système de CEFSMs. Dans ce cas, une CEFSM ne peut être appelée qu'une fois

et une seule dans le système de CEFSMs. La CEFSM appelée est alors composé d'un seul chemin. Cette stratégie de composition est déjà accomplie par l'étape *Analyse de Traces et Extraction*.

**Composition faible :** l'intuition de cette stratégie consiste à regrouper les CEFSMs ayant une forte similarité. En effet, les CEFSMs similaire doivent provenir du même composant. La notion de similarité est définie et évaluée par un coefficient de similarité. Ce coefficient est défini par le recouvrement (Overlap) des symboles et des paramètres entre 2 CEFSMs. Nous construisons une matrice de similarité à partir des coefficients, ce qui nous permet d'utiliser de nombreux algorithmes de clustering pour trouver les classes de CEFSMs similaires, qui sont assemblées dans le même cluster. Les CEFSMs dans le même cluster sont fusionnées via une union disjointe. Toute les transitions $s_1 \xrightarrow{call(CEFSM:=C_i)} s_2$ sont mise à jour pour que la bonne CEFSM soit appelée, en remplaçant le $C_i$ par la CEFSM correspondant au cluster. De plus, toutes les transitions $s_1 \xrightarrow{call(CEFSM:=C_i)} s_2$ sont remplacées par une boucle $(s_1, s_2) \xrightarrow{call(CEFSM:=C_i)} (s_1, s_2)$, en fusionnant les états $s_1$ et $s_2$.

**Composition forte :** cette stratégie est basée sur la précédente, à laquelle nous rajoutons en plus pour tout état $s$, une transition $s \xrightarrow{call(CEFSM:=C_i)} s$ pour chaque CEFSMs $C_i$. Les CEFSMs deviennent maintenant callable-complete.

Dans la section suivante, nous montrons un exemple d'intégration avec la méthode passive GK-tail [3].

# 4 Intégration avec GK-tail

GK-tail est une méthode d'inférence de modèle passive proposée par Lorenzoli et al. [3], inférant une EFSM à partir de plusieurs traces d'exécution d'un système, et ceci en quatre étapes : la fusion de traces, la génération de prédicat, la construction d'une première EFSM, et enfin la fusion d'états équivalents.

Notre approche COnfECt peut s'intégrer à GK-tail comme décrit en Figure 2. L'étape *Analyse de Traces et Extraction* de COnfECt permet de segmenter les traces et d'identifier des composants. Elle est réalisée après la fusion des traces (étape 1 GK-tail), pour accélérer le calcul du coefficient, car moins de traces sont à analyser. L'étape *Composition de CEFSMs* permet de grouper certains CEFSMs similaires, si la stratégie composition faible ou forte est employée. Cette étape est faite avant que GK-tail ne fusionne les états. Cet ordre peut favoriser le groupement d'état équivalents qui seront fusionnés ensuite.

La Figure 3 illustre un exemple de ce que nous pouvons obtenir en utilisant notre méthode avec GK-tail sur un petit échantillon de traces d'objet connecté. L'objet est composé d'une interface WEB, d'un capteur de température, ainsi qu'un capteur de mouvement. Notre méthode permet de séparer d'un coté l'interface WEB, à gauche dans la Figure 3, et les capteurs à droite.

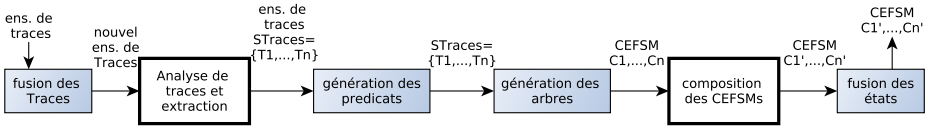


FIGURE 2 – Intégration des étapes avec GK-tail.

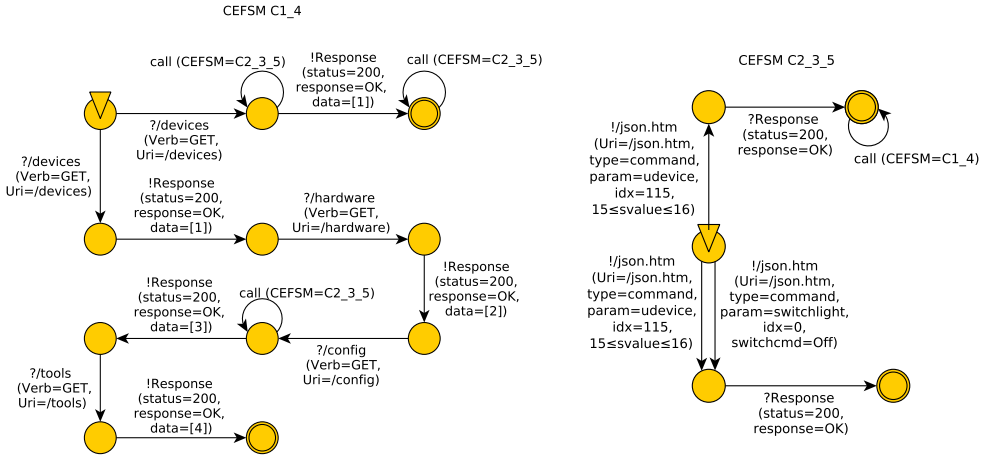


FIGURE 3 – Exemple de CEFSMs obtenus.

# 5 Conclusion

Nous avons introduit la méthode COnfECt, qui permet d'inférer un système de CEFSMs à partir de traces d'exécution d'une implémentation. Des composants sont identifiés dans les traces, et sont modélisés par des CEFSMs, qui peuvent être appelés à partir d'autres CEFSMs. Les algorithmes et définitions sont disponibles dans [2]. En termes de perspectives, il faudrait adapter la méthode afin de l'utiliser sur des systèmes aux communications asynchrones, ou encore considérer des composants dont les exécutions se font en parallèle. Nous prévoyons l'implémentation de cette méthode et son intégration avec GK-tail pour effectuer des évaluations.

# Références

[1] D. ANGLUIN : Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87 – 106, 1987.

[2] E. BLOT, S. SALVA et P. LAURENÇOT : COnfECt : Une Méthode Pour Inférer Les Modèles De Composants D'un Système. Research report, fév. 2018. http ://sebastien.salva.free.fr/RR-18-02.pdf.

[3] D. LORENZOLI, L. MARIANI et M. PEZZÈ : Automatic generation of software behavioral models. *In Proceedings of the 30th International Conference*

*on Software Engineering*, ICSE '08, p. 501–510, New York, NY, USA, 2008. ACM.