COnfECt : A Passive Model Learning Method For Component-based Systems

Elliott Blot:
Limos – UMR CNRS 6158, Clermont Auvergne University. France
elliott.blot@uca.fr

Ecole doctorale Sciences Pour l'Ingénieur

Introduction

A behavioural model of a system is useful to help engineers understand how a system is functioning and audit it. However, create such a model by hands is a long and error prone task. *Model learning* methods are methods that infer a behavioural model of a system. There exist two types of model learning methods :

- *Active methods* [1], query directly the system or the user to obtain observations, used to build or improve a model. This type of method does not work on every system, like uncontrollable systems or systems that cannot be reset like IoT systems.
- *Passive methods* [2], deduce a model from data extracted from the system like for example execution traces. These methods can return incomplete model, and huge model difficult to understand.

IoT systems are generally composed of many reusable components, and model these components separately can lead to smaller and more understandable models.

COnfECt (COrrelate Extract Compose) is a passive model learning method that generates model of each component of the system from execution traces of the system. A more detailed presentation of this work can be found in [3]

The COnfECt Method

We suppose that the events of the log have the following form : *<label>(parm1, param2,…).*

```
e1:  /devices (Verb, Uri)
e2:  /json.htm (param, svalue)
e3:  Response (response)
e4:  Response (response, data)
e5:  /json.htm (param, svalue)
e6:  Response (response)
e7:  /devices (Verb, Uri)
e8:  Response (response, data)
e9:  /hardware (Verb, Uri)
e10: Response (response, data)
e11: /config (Verb, Uri)
e12: /json.htm (param, switchcmd)
e13: Response (response)
e14: Response (response, data)
e15: /tools (Verb, Uri)
e16: Response (response, data)
```

Figure 1. Example of formatted traces.

The COnfECt method is separated into 4 steps : Trace Extraction, LTS Generation, LTS Synchronization, and state merging.

STEP 1 : Trace Extraction:
In this step, we try to separate the events that come from different components. For that we use a *correlation coefficient*, used to determine if two events in a trace come from the same component. This coefficient is defined by the user and depends of the system, for example, it can be based on frequency of the events. We use this coefficient to separate each trace into sequences of consecutive events with a high correlation. Then, we remove of the trace each sequence that come from an other component than the first event of the trace, and replace it by synchronization event *call_Ci* and *return_Ci*.

```
Trace T1:
e1:  /devices (Verb, Uri)
     call_C2
     return_C2
e4:  Response (response, data)
     call_C3
     return_C3
e7:  /devices (Verb, Uri)
e8:  Response (response, data)
e9:  /hardware (Verb, Uri)
e10: Response (response, data)
e11: /config (Verb, Uri)
     call_C4
     return_C4
e14: Response (response, data)
e15: /tools (Verb, Uri)
e16: Response (response, data)

Trace T2:
     call_C2
e2:  /json.htm (param, svalue)
e3:  Response (response)
     return_C2

Trace T3:
     call_C3
e5:  /json.htm (param, svalue)
e6:  Response (response)
     return_C3

Trace T4:
     call_C4
e12: /json.htm (param, switchcmd)
e13: Response (response)
     return_C4
```

Figure 2. Trace Extraction Step.

STEP 2 : LTS Generation:
In this step, we generate for each the trace *Ti*, a LTS *Ci* with only one path that correspond of the trace, where each event is modelled by a transition.

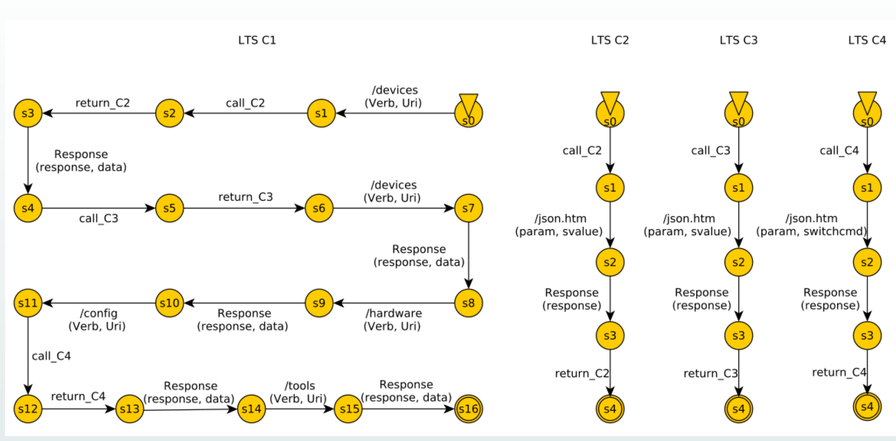

Figure 3. LTSs generated from the four previous traces.

STEP 3 : LTSs Synchronization:
Different synchronization strategies are proposed for our models. Here only the *weak* strategy is showed, where we merge the similar LTSs that come from the same component, with the help of clustering technique and generalize the models by letting a component call an other many times in a row (modelled by a loop *call_Ci return_Ci*).

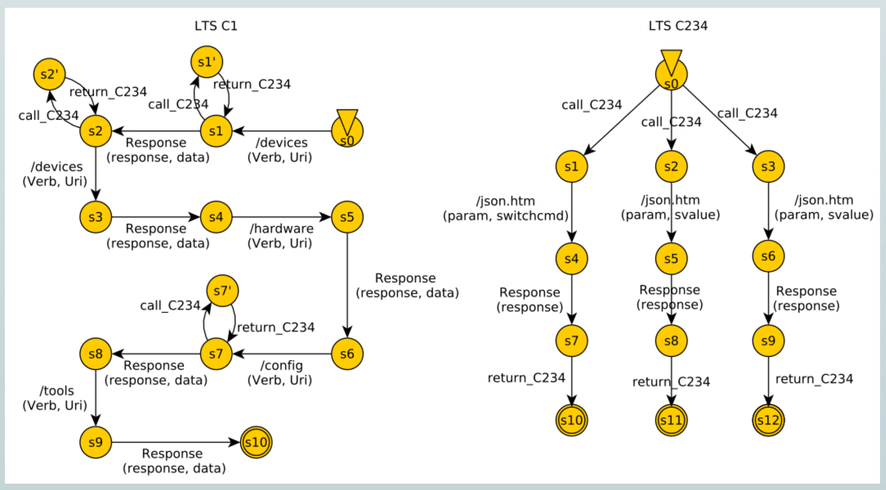

Figure 4. Generalization of the *weak* strategy of COnfECt.

STEP 4 : State Merging:
In this step we merge the equivalent states with the help of kTail[2] with k=2, that merges states that have the same future of length k.

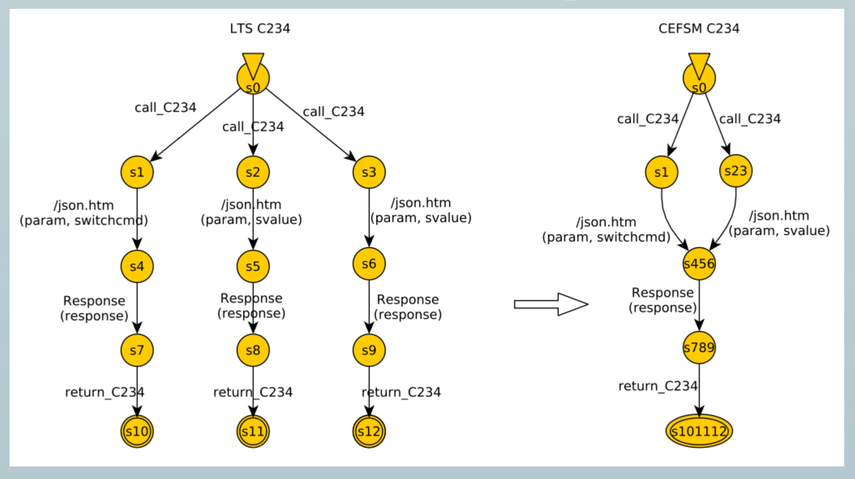

Figure 5. Result of kTail on the LTS C234 (C1 stay unchanged).

Conclusion

The COnfECt method can generate the model of the different component of the system. These models can be be used to analyze the whole system, with the help of the synchronization events *call_Ci* and *return_Ci*, or to analyze a specific component of the system by hiding these events.
In term of perspective, we plan to use these models, to verify some security property.

Bibliography

1. **Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation,* 75(2):87-106, 1987**
2. **A.W. Biermann and J.A. Feldman. On the synthesis of finite-state machines from sample of their behavior. *Computers, IEEE Transactions,* 592-597, June 1972**
3. **Sébastien Salva, Elliott Blot. COnfECt: An Approach to Learn Models of Component-based Systems. ICSOFT 2018: 298-305**