Reverse Engineering Models of Concurrent Communicating Systems From Event Logs

Sébastien Salva (sebastien.salva@uca.fr)

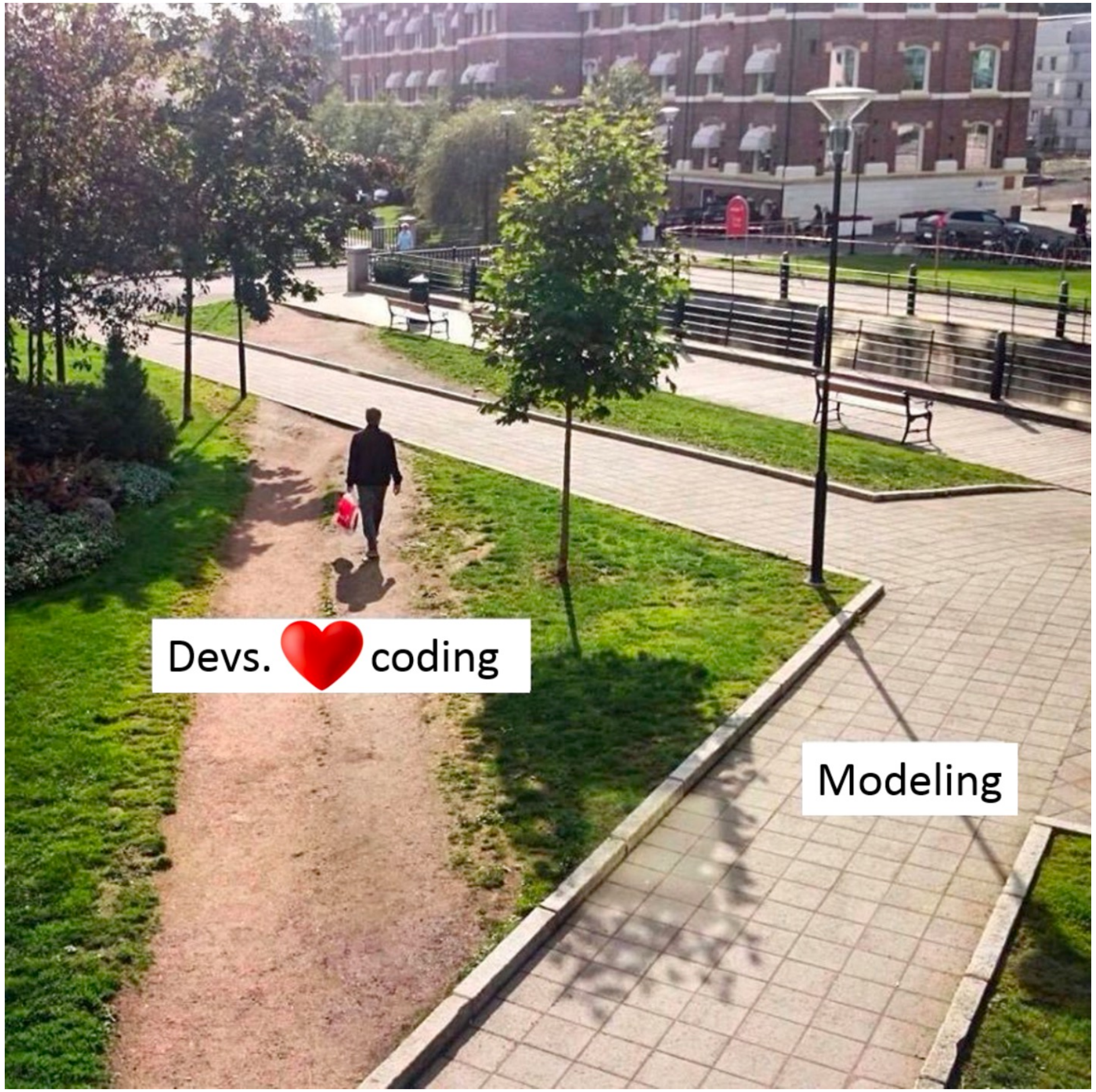
LIMOS, Clermont Auvergne University, France

Sixteenth International Conference on Software Engineering Advances, oct. 2021

Who am I?

```
Public void setUp(){
Identity id=new Identity(“salva’’);}

Public void testid (){
assertEquals(id.surname, ’’sébastien’’);
assertEquals(id.name, ’’salva’’);
assertEquals(id.labo, ’’LIMOS’’);
assertEquals(id.univ ’’University Clermont Auvergne, France’’);

assertArrayEquals(i.recherche, new String[] {“testing’’, “security’’, ’’model learning’’,
“services“});
}
```

Introduction



Introduction

Model Learning :

- Generation of behavioral models from a black box application (by retro-engineering).

Use of Models : for documentation, analysis, auto generation of test cases, etc.

Several limitations:

- Need of Working app. or execution traces
- Extract accurate conversations (a.k.a. sessions) when applications are made up of concurrent components
- Modeling behaviours of every component
- Avoid spaghetti models (unredable and large models)

Introduction

Assumptions :

- System under learning = Concurrent Communicating Systems made up of components
- No knowledge of the components
- correlation mechanisms are employed to propagate context IDs and keep track of the process contexts among components
- But, we don’t know them

Proposal:

- Passive model learning approach and tool to recover Input Output Labelled Transition Systems (IOLTSs) from event logs.
- Algorithm to automatically retrieve conversations from event logs, without having any knowledge about the used correlation mechanisms.

Paper presentation

1. Overview
2. Model learning from logs of IOT systems
3. Choice of the components to mock
4. Mock generation/execution
5. (preliminary) Evaluation

Big Picture

Event Log → **Correlaton Ids & Conversation Extraction** → **Trace Partitionning** → **IOLTS Generation** → **IOLTS Generalisation**

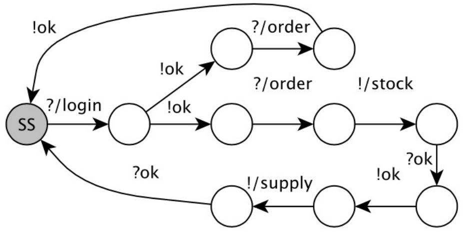
```
/login(from:="cl", to:="ShopS", id:="tocken",
    account:="1")
ok(from:="ShopS", to:="cl", id:="tocken"
    trans:="t1")
/order(from:="cl", to:="ShopS",
    trans:="t1",item:="a")
/stock(from:="ShopS", to:="StockS",
    trans:="t1", item:="a")
ok(from:="StockS", to:="ShopS", trans:="t1",
    item:="a")
ok(from:="ShopS", to:="cl",
    trans:="t1",content:="stock")
/supply(from:="ShopS", to:="WS", trans:="t1",
    key:="k1",item:="a")
ok(from:="WS", to:="ShopS", trans:="t1",
    key:="k1")
/supplyWS(from:="WS", to:="WS1", key:="k1",
    key2:="k2",item:="a")
/supplyWS(from:="WS", to:="WS2", key:="k1",
    key2:="k3",item:="a")
/supplyWS(from:="WS", to:="WS3", key:="k1",
    key2:="k4",item:="a")
ok(from:=WS1,to:="WS", key:="k1", key2:="k2")
ok(from:=WS2,to:="WS", key:="k1", key2:="k3")
Unavailable(from:="WS3",to:="WS", key:="k1",
    key2:="k4")
/login(from:="cl", to:="ShopS", id:="tocken2",
    account:="12")
ok(from:="ShopS", to:=cl, id:="tocken2",
    trans:="t2")
/order(from:="cl", to:="ShopS",
    trans:="t2",item:="b")
ok(from:="ShopS", to:="cl", trans:="t2"
    content:="no stock")
```

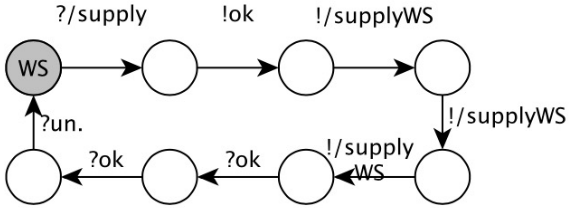
Corr keys
:={id,key,key2}

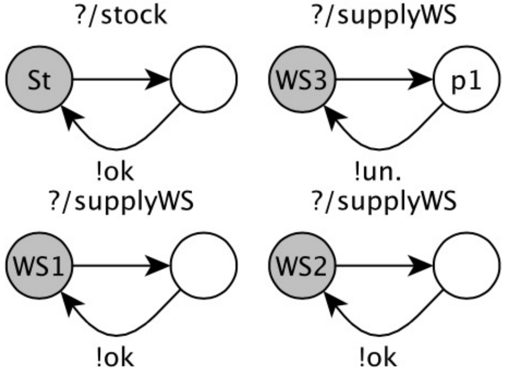
Conv:={
/login() ok()
/order() /stock()
ok() ok()/supply()
ok()
/supplyWS()/supp
lyWS()
/supplyWS() ok()
ok() unavailable(),

/login() ok()
/order() ok()
}

5 components
5 trace sets:
T(SS)
T(WS)
T(St)}
T(WS1)
T(WS2
T(WS3







Conversation Extraction

How to find Correlation keys ?

- Using a brute- force search ? No time consumming, not effiscient

- Our proposal: algorithm is based upon a formalisation of the notion of correlation patterns and guided towards the most relevant conversation sets by evaluating conversation quality.

Conversation Extraction

Correlation patterns :

- Key based correlation:
  - /login(from:="cl", to:="ShopS", id:="tocken", account:="l") ok(from:="ShopS", to:="cl", id:="tocken" trans:="t1")
- Chained correlation:
  - /order(from:="cl", to:="ShopS", trans:="t1",item:="a")
- Function based correlation ans Time-based correlation
  - /login(t:= 1) ok(t:=1) /order(t:=1) with t= floor(time(eevent)/T, T=5s

Conversation Extraction

From Correlation patterns :

- Extraction of conversation invariants = properties on conversations and correlation key sets that must hold
- Definition of 4 Conversation Quality metrics = metrics between 0 and 1 to express conversation structure with regard to inputs, outputs etc.

Conversation Extraction

Conversation quality metric example :

$$0 < m_1(\sigma) = \frac{|ReqwResp(\sigma)| + 1}{|Req(\sigma)| + 1} \leq 1 \qquad (1)$$

m1 evaluates the ratio of requests iassociated to some responses

$$0 < m_2(\sigma) = \frac{|RespwReq(\sigma)| + 1}{|Resp(\sigma)| + 1} \leq 1 \qquad (2)$$

m2 measures the ratio of responses following a prior request

Conversation Extraction

- In a nutshell :

1. Coverage of the succesive events of an event logs and serach of potential correlation keys
2. Computation of invariants and quality metrics
3. If invarants do not hold or low quality stop

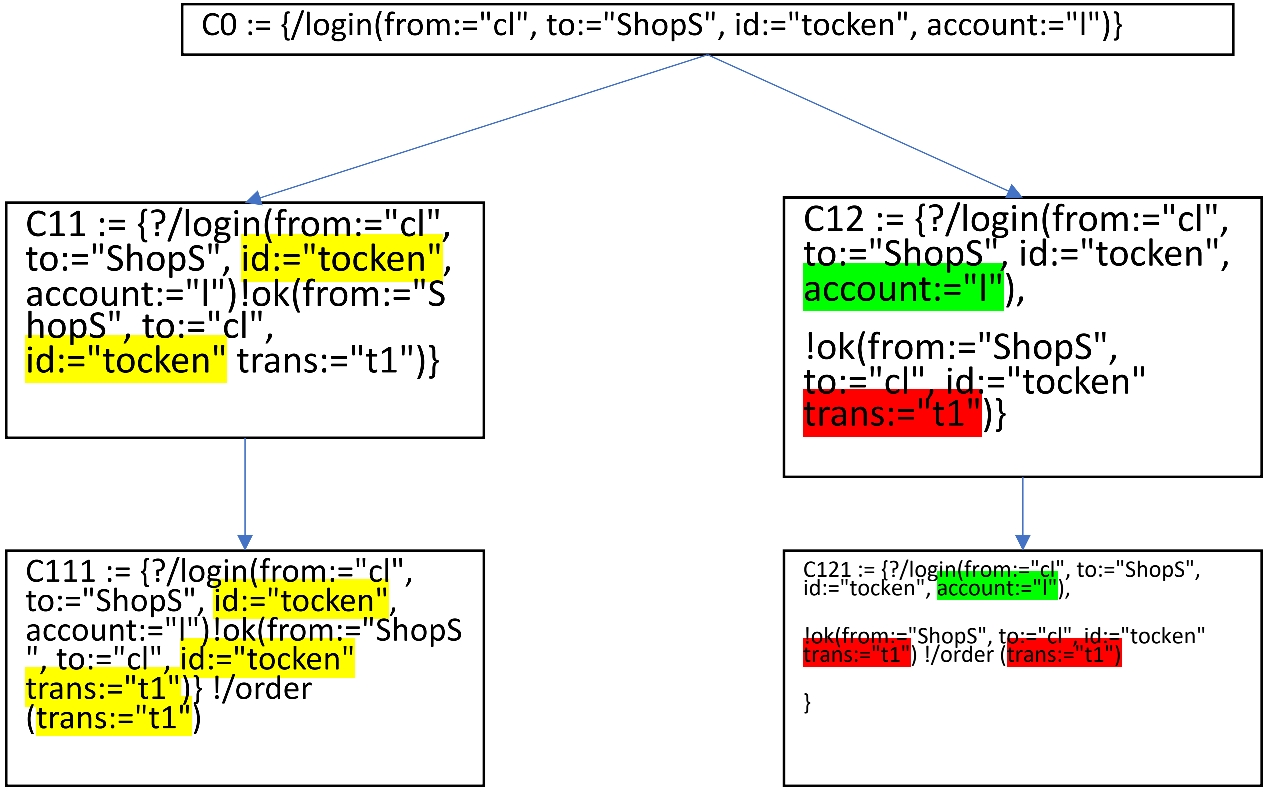
Conversation Extraction

/login(from:="cl", to:="ShopS", id:="tocken", account:="l")

ok(from:="ShopS", to:="cl", id:="tocken" trans:="t1")

/order(from:="cl", to:="ShopS", trans:="t1",item:="a")

/stock(from:="ShopS", to:="StockS", trans:="t1", item:="a")



requests followed by responses
Good quality

/login alone ?
Response followed by request ?
Lower quality

Conversation Extraction

```
/login(from:="cl", to:="ShopS", id:="tocken",
    account:="l")
ok(from:="ShopS", to:="cl", id:="tocken"
    trans:="t1")
/order(from:="cl", to:="ShopS",
    trans:="t1",item:="a")
/stock(from:="ShopS", to:="StockS",
    trans:="t1", item:="a")
ok(from:="StockS", to:="ShopS", trans:="t1",
    item:="a")
ok(from:="ShopS", to:="cl",
    trans:="t1",content:="stock")
/supply(from:="ShopS", to:="WS", trans:="t1",
    key:="k1",item:="a")
ok(from:="WS", to:="ShopS", trans:="t1",
    key:="k1")
/supplyWS(from:="WS", to:="WS1", key:="k1",
    key2:="k2",item:="a")
/supplyWS(from:="WS", to:="WS2", key:="k1",
    key2:="k3",item:="a")
/supplyWS(from:="WS", to:="WS3", key:="k1",
    key2:="k4",item:="a")
ok(from:=WS1,to:="WS", key:="k1", key2:="k2")
ok(from:=WS2,to:="WS", key:="k1", key2:="k3")
Unavailable(from:="WS3",to:="WS", key:="k1",
    key2:="k4")
/login(from:="cl", to:="ShopS", id:="tocken2",
    account:="l2")
ok(from:="ShopS", to:=cl, id:="tocken2",
    trans:="t2")
/order(from:="cl", to:="ShopS",
    trans:="t2",item:="b")
ok(from:="ShopS", to:="cl", trans:="t2"
    content:="no stock")
```

Corr keys :={id,key,key2}

2 conversations :
Conv:={
/login() ok() /order() /stock() ok() ok()/supply() ok()
/supplyWS()/supplyWS() /supplyWS() ok() ok() unavailable(),

/login() ok() /order() ok()
}

Trace partionning

Detection of components with analysis of emitter and receivers in events

Generation of as many trace sets as components found

Algorithm to segment the conversations into sub-sequences, each capturing the behaviours of one component only

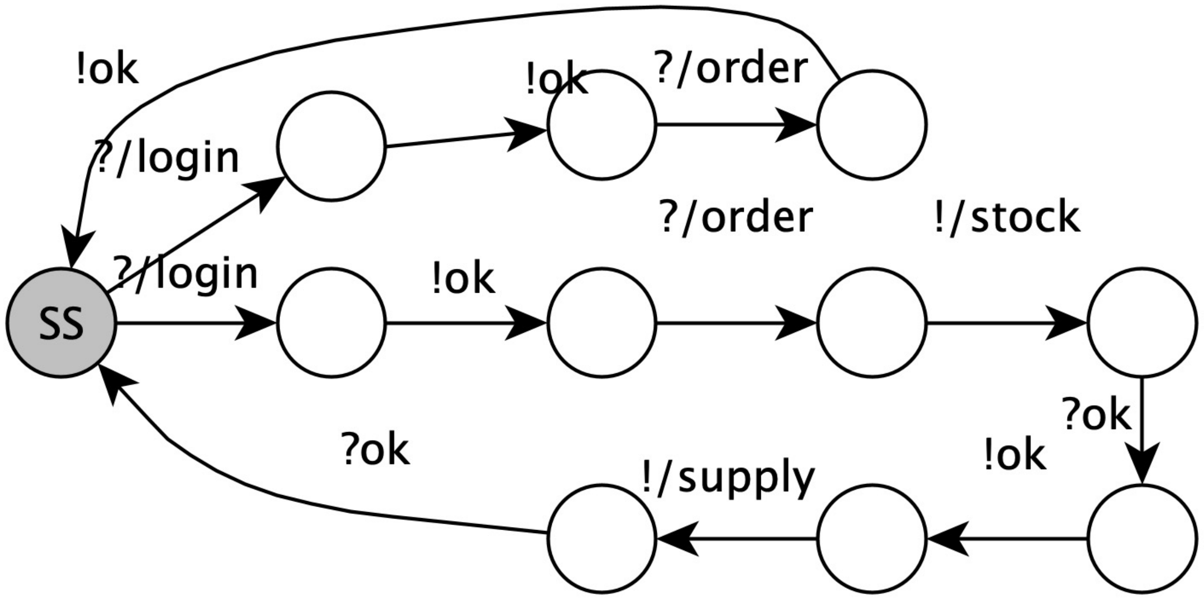
```
/login(from:="cl", to:="ShopS )
/order(from:="cl", to:="ShopS")
/stock(from:="ShopS",
to:="StockS") ok(from:="StockS",
to:="ShopS") ok(from:="ShopS",
to:="cl").....
```

```
T(ShopS)={ ?/login() !ok() ?/order() !/stock() ?ok() !ok()
!/supply() ?ok(), ?/login() !ok() ?/order()  !ok() }

T(StockS)={ ?/stock() !ok() }

T(WS)={ ?/supply() !ok() !/supplyWS() !/supplyWS()
!/supplyWS() ?ok() ?ok() ?unavailable() }

T(WS1)=T(WS2)={ ?/supplyWS() !ok() }

T(WS3)={ ?/supplyWS() !unavailable() }
```

IOLTS Generation

Every trace set lifted to the level of IOLTS
1 trace set –> 1 IOLTS obtained by transforming the traces IOLTS paths having the same initial state only

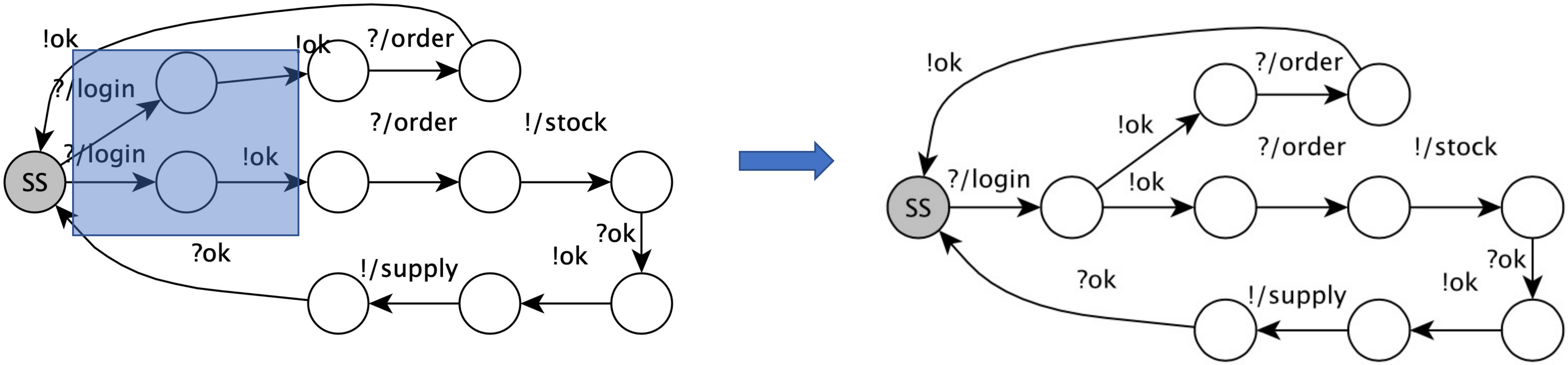
T(ShopS)={ ?/login() !ok() ?/order() !/stock() ?ok() !ok() !/supply() ?ok(), ?/login() !ok() ?/order() !ok() }



IOLTS Generalisation

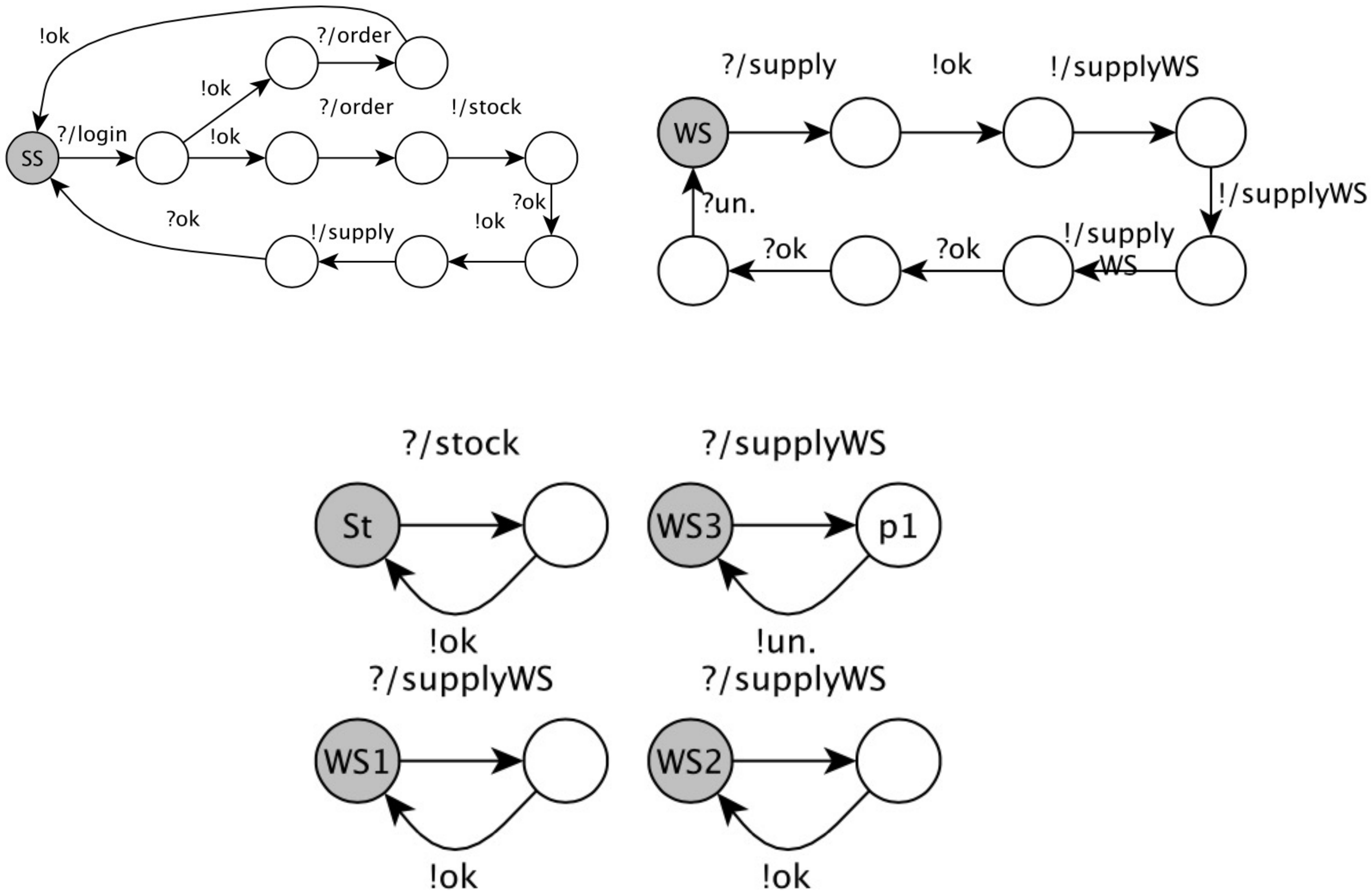
IOLTS generalisation by merging their equivalent states.
State merging performed by means of the k-Tail algorithm

> assembles the states sharing the same k-future, i.e., the same event sequences having the maximum length k

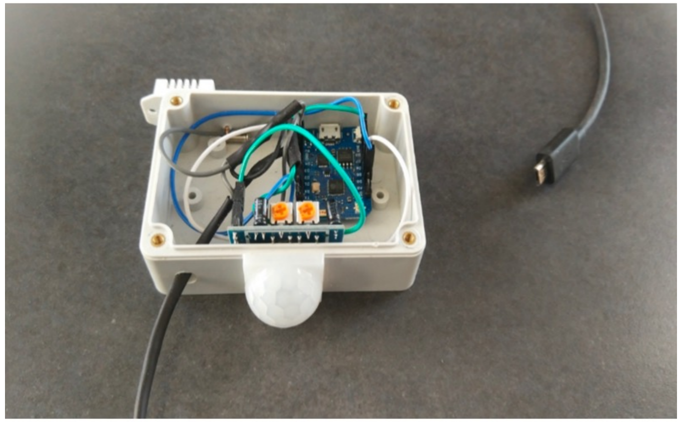


Exemple: Final Results

```
/login(from:="cl", to:="ShopS", id:="tocken",
    account:="l")
ok(from:="ShopS", to:="cl", id:="tocken"
    trans:="t1")
/order(from:="cl", to:="ShopS",
    trans:="t1",item:="a")
/stock(from:="ShopS", to:="StockS",
    trans:="t1", item:="a")
ok(from:="StockS", to:="ShopS", trans:="t1",
    item:="a")
ok(from:="ShopS", to:="cl",
    trans:="t1",content:="stock")
/supply(from:="ShopS", to:="WS", trans:="t1",
    key:="k1",item:="a")
ok(from:="WS", to:="ShopS", trans:="t1",
    key:="k1")
/supplyWS(from:="WS", to:="WS1", key:="k1",
    key2:="k2",item:="a")
/supplyWS(from:="WS", to:="WS2", key:="k1",
    key2:="k3",item:="a")
/supplyWS(from:="WS", to:="WS3", key:="k1",
    key2:="k4",item:="a")
ok(from:=WS1,to:="WS", key:="k1", key2:="k2")
ok(from:=WS2,to:="WS", key:="k1", key2:="k3")
Unavailable(from:="WS3",to:="WS", key:="k1",
    key2:="k4")
/login(from:="cl", to:="ShopS", id:="tocken2",
    account:="l2")
ok(from:="ShopS", to:=cl, id:="tocken2",
    trans:="t2")
/order(from:="cl", to:="ShopS",
    trans:="t2",item:="b")
ok(from:="ShopS", to:="cl", trans:="t2"
    content:="no stock")
```



(preliminary) Evaluation



Conducted on 6 IoT systems integrating varied devices and gateways communicating over HTTP and UDP

- RQ1: Can The Approach Extract Relevant Conversation Sets?
- RQ2: what is the performance of our algorithm?

(preliminary) Evaluation

RQ1: Can The Approach Extract Relevant Conversation Sets?

Event logs ~ 2200 events

Manually analysed the event logs S1 to S6 and computed Precision and Recall on the generated conversations

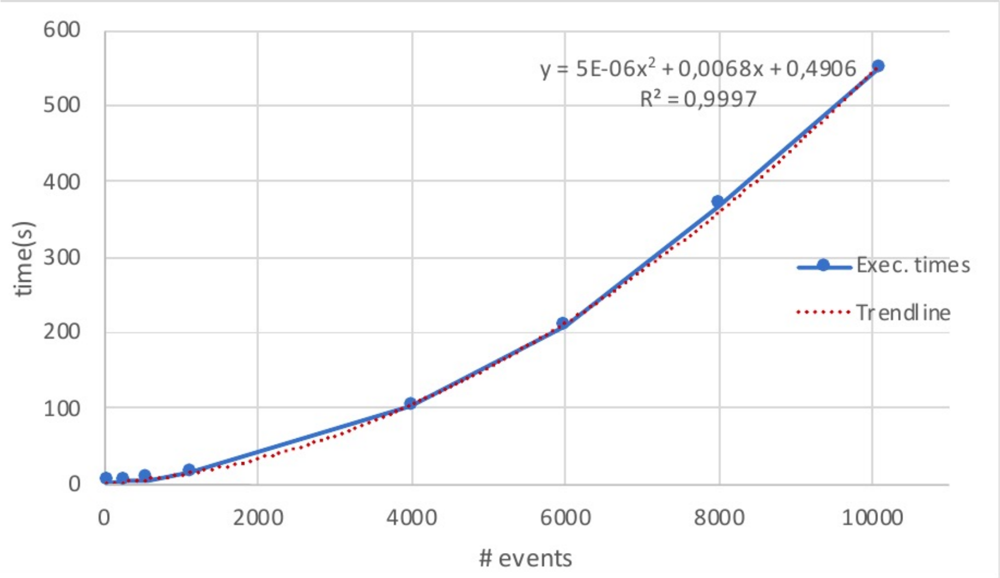
	Correlation Key Set Recall	Correlation Key Set Precision
*S*1	100%	81%
*S*2	100%	76%
*S*3	100%	80%
*S*4	100%	100%
*S*5	100%	100%
*S*6	100%	90%

Results: provides good recall and precision, but sometimes returns several results, choice can be conducted with quality metrics ordering

(preliminary) Evaluation

- RQ2: what is the performance of our algorithm?

1. We took the 20 first conversations of S1 and augmented them using 40 to 10000 events;

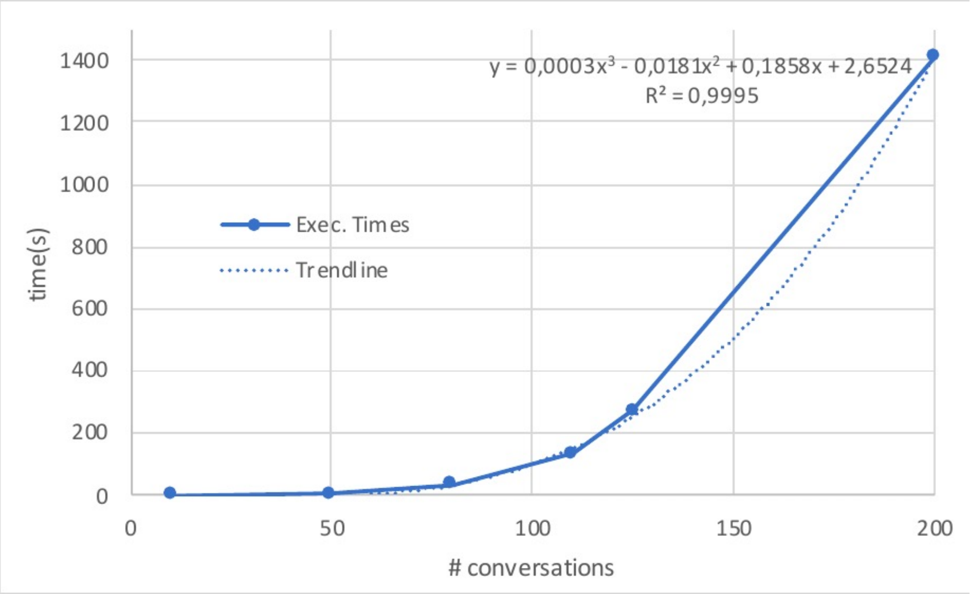


The curve follows a quadratic curve and reveals that our approachperforms well in practice.

(preliminary) Evaluation

- RQ2: what is the performance of our algorithm?

2. We measured execution times with regard to the number of conversations in the event logs from 10 to 200 conversations of 2 events.



cubic polynomial curve, which shows that execution times quicker increase with regard to the number of conversations.

Conclusion

- Design of a model learning approach specialised into the recovery of formal models from event logs generated by communicating systems made up of concurrent components
- The generated IOLTS can be later used as documentation or for automatics analyses (security testing, etc.)

Limitations:

- Need of a good balance between model size, readability and precision. For instance, the generated IOLTSs may be very large on account of similar event sequences having different parameter values.

Thanks

- Questions ?