Second Sprint

Streaming event data compliance checking in Python

Jingjing Huo	376323
Sabya Shaikh	384606
Zheqi Lyu	378653

1 SetUp

1.1 Configuration

Changing User-Defined parameters on Server side via manipulating the content in the file - config.ini (under project folder), there are following parameters:
Window Size: the default value is [1,2,3,4], and one can change it into whatever he wants to have, like [1,2,3,4,5,6] or [2,4,6] etc.
Threshold: the default value is 0.2.
Path of the event log for training: the default path is 'data/Example_EventLogForTraining _Backup.xes'. If one wants to change it, the file should under the project folder.
Checking type: The type of compliance checking:

1. Option: DELETE_M_EVENT: This means when one event is detected as a spurious event, if the reason is it is not in the automata(the alert type is 'M'), we delete this event from the corresponding case, then do the following checking.
2. Option: KEEP_ALL_EVENTS(default): This means that we keep all events during the compliance checking despite being spurious or not.

Alert type: The type of response of alert: (Not implement yet!)

1. Option: RETURN_ONE(default): If one event alerts during checking with window size 1, we don't continue doing checking with bigger window size, and only return one alert into the client, because in this case, this event will obviously alert when checking with bigger window size.
2. Option: RETURN_ALL_ALERTS: This means when one event is spurious, we will return all alerts for different window size.

2 Source Code description

2.1 Client - Improvement

User Friendly:

If the client has done the compliance checking, they will get to the warning, if they really want to do again.

```
[20:58:0:249137]Warning : You have already done the compliance check! Do you rea
lly want to restart? Or do you want to render the deviation pdf?
        If you want to restart, please press 1 again!
        If you want to skip, please press 2!
```

Concurrency Avoiding:

The clients with same user name are not allowed to do the compliance checking at the same time.

```
[20:25:17:378172]Refuse : The user with the same name is just doing the complian
ce checking, please try it with other name!
```

Response Format:

We have two kinds of alert type: Missing(M) and Low Threshold(T).
The structure of the responses that will be sent by server is as below:

json:{
'case_id': event['case_id'],
'source_node': source_node,
'sink_node': sink_node,
'expect': {expect_sink_node, probability of this connection},
'body': 'M'
}

json:{
'case_id': event['case_id'],
'source_node': source_node,
'sink_node': sink_node,
'cause': the true threshold providing by server configuration
'expect': THRESHOLD,
'body': 'T'
}

The results that the client will be received show in the following picture:

```
[21:9:35:758539]Alert M :  no such connection in case '174138'
    The connection: App_Fully_Submission --> Err_Incomplete_Submission
    The expected connection:
         App_Fully_Submission --> App_Incomplete_Submission :  1.0
[21:9:35:763304]Alert M :  no such connection in case '174138'
    The connection: Err_Incomplete_Submission --> W_Handling_Leads
[21:9:35:768127]Alert M :  no such connection in case '174138'
    The connection: Err_Incomplete_Submission,W_Handling_Leads --> W_Handling_Leads,W_Handling_Leads
[21:9:35:773797]Alert M :  no such connection in case '174138'
    The connection: Err_Incomplete_Submission,W_Handling_Leads,W_Handling_Leads --> W_Handling_Leads,W_Handling_Lea
ds,App_Pre_Acceptation
[21:9:35:779308]Alert M :  no such connection in case '174138'
    The connection: Err_Incomplete_Submission,W_Handling_Leads,W_Handling_Leads,App_Pre_Acceptation --> W_Handling_
Leads,W_Handling_Leads,App_Pre_Acceptation,W_Complete_Application
[21:9:35:784113]Alert T :  The threshold of the connection in case '174138' is too low.
   The minimal expected probability from  W_Complete_Application --> W_Handling_Leads :  0.2
              The true probability from  W_Complete_Application --> W_Handling_Leads :  0.0316095
```

Console View:

We use console. logging to print the information, which can distinguish the different type of information. In that way, the client can catch the information more quickly.

```
(StreamEventCompliance) 172-015:client xuer$ python client.py client1 Example_EventLogForTesting.xes
[21:56:22:402204][  ERR  ] : ConnectionError: The server is not available, please try it later!
```

```
[21:9:35:823341][ INFO  ] : ------------------the compliance checking is finishing----------------------
There are two services:
        Press 1, if you want to do the compliance checking
        Press 2, if you want to show the deviation pdf
        Press 3, if you want to exit
[21:9:35:823780]Note: : you can interrupt with CTR_C, once you start to do the compliance checking
[21:9:36:3360][ INFO  ] : Info:The compliance checking is over, you can get the deviation pdf!
```

2.2 Server

2.2.1 Speedup using Hash-Table

Instead of using LIST to store the different connections between the nodes to do compliance checking, we use DICT with HASH. The runtime has been sufficiently speeded up, particularly by large file. We try to run the Eventlog with 99M. The runtime of building automata is reduced from around 130s to 70s.

2.2.2 Compliance Checking

2.2.2.1 Compliance checking details

For each event sent to server, we do a compliance checking.
Firstly, we add it to the case memory, then we calculate the windows memory for it.
For example, if incoming event “a” is the first event for a case_id and maximum window size is 4 then the windows memory will be [*, *, *, *, 'a'].
As the events keep coming, we move add the latest event at 5th position and move the previous events in windows memory by 1 to the left.
For eg, if we now receive “b” the windows memory will be [*, *, *, ‘a’, 'b']

After calculating windows memory for each event we calculate the sink_node and source_node using this windows memory.

	Windows Memory		[*,*,*,’a’, ‘b’]	[a,b,c,d,e]
1	Windows size = 1	source_node	a	d
		sink_node	b	e
2	Windows size = 2	source_node	None	c,d
		sink_node	a,b	d,e
3	Windows size = 3	source_node	-	b,c,d
		sink_node	-	c,d,e
4	Windows size = 4	source_node	-	a,b,c,d
		sink_node	-	b,c,d,e

We create a Connection object using the source_node and sink_node calculated above.
This connection object is checked in ***autos*** object which was created while training automata(automata is preloaded into computer memory in autos object for speeding up).
Autos object is the automata with varying prefix length.

autos= {

1 : { window size: 1,nodes: {node: ‘’, degree: ‘’}, connection_list : “list”} ,
2 : { window size: 2,nodes: {node: ‘’, degree: ‘’}, connection_list : “list”} ,
3 : { window size: 3,nodes: {node: ‘’, degree: ‘’}, connection_list : “list”} ,
4 : { window size: 4,nodes: {node: ‘’, degree: ‘’}, connection_list : “list”} ,
}

.

Case 1: source_node is not None:

The automata’s connection list is scanned to check if newly created connection object is available in it. If available then probability of this connection is checked. The threshold of the probability is set to 0.2. If the connection exists and probability is equal or greater than threshold then the event is considered as normal event. If the connection is not available in automata or probability is lesser than threshold then it is considered to be spurious event and alert is raised.

Case 2: source_node is None:

We check if sink_node exists in nodes dictionary. This indicates that sink_node is the starting node (starting events). If sink_node does not exists alert is raised.

Alert details are inserted into ***alertlog*** object and corresponding response is sent to client. Alertlog object is inserted in ***alertrecord*** table once all the events are processed from same client.

“alertrecord” table contains

Column	Data type	Value
user_id	varchar(100)	client_uuid
source_node	varchar(100)	
sink_node	varchar(100)	
alert_cause	varchar(1)	T/M
alert_count	float	

Alert_cause **T** indicate that the alert was due to probability of connection being lower than threshold

Alert_Cause **M** indicate that there is no connection with that source_node and sink_node

Response sent to client is of the json form:

```
{
    'case_id':
    'source_node':
    'sink_node':
    'cause':
    'expect':
    'body':
}
```

The body is set to “M” or “T” or “OK”
Body set to “OK” indicates there is no deviations for that event
The client side handles the response based on body.

If windows memory is [a,b,c,d,e] after arrival of spurious event “e”
For window size=1 -> source_sink = d, sink_node = e -------Alert will be raised
For window size=2 -> source_sink = c,d, sink_node = d,e -------Alert will be raised
For window size=3 -> source_sink = b,c,d, sink_node = c,d,e -------Alert will be raised
For window size=4 -> source_sink = a,b,c,d, sink_node = b,c,d,e -------Alert will be raised
Alerts will be raised until e is in the windows_memory

Here 4 alerts will be raised for one spurious event. These are redundent alerts.
Hence, we aim at providing two options in case the the connection is not found in automata as described in Configuration, Option Alert Type .

2.2.2.2 Muliti-Threading & Lock for Case and Alerts Updating

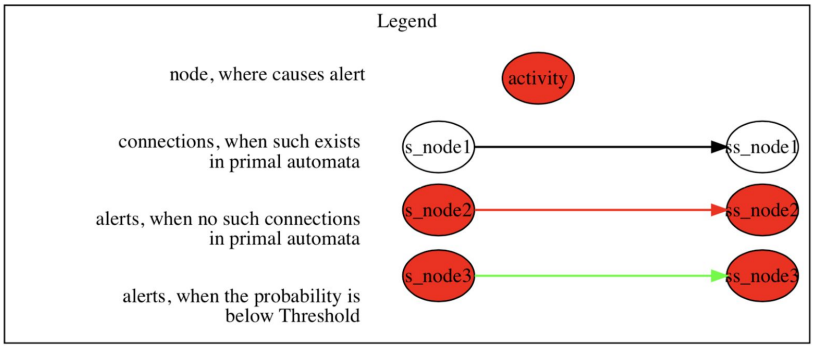
The implementation of this part is similar to the multi-threading part for building automata, the most different thing is we need to keep the separation of cases from different clients. In such a case, we can do several compliances checking for different clients simultaneously. To this end, we change the structure as follows:

CaseMemorizer.dictionary_cases= {
‘uuid1’ : { case_id1 : [* , * , * , * , ‘a’], case_id2 : [* , * , * , ‘b’, ‘c’], ... },
‘uuid2’ : { case_id1 : [* , * , * , * , ‘a’], case_id2 : [* , ‘a’, ‘r’, ‘p’, ‘q’], ...}, ...
}

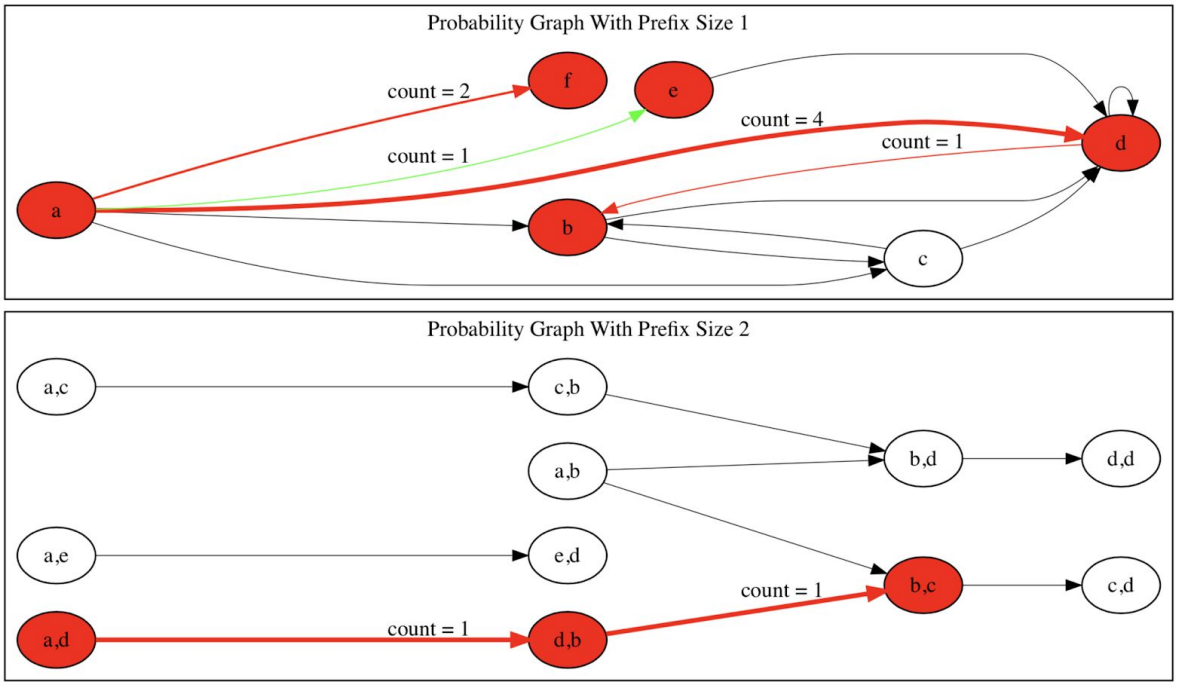
Both case lock and alert lock are based on the single case or single alert of a client, no interactions of cases or alerts from different clients. This guarantees that the compliance checking runs for different clients parallelly.

2.2.3 Generate Alert Automata PDF

We use different colors to refer to the different types of alert and the thickness of the line to indicate the counts of the alert in that connection. Legend for the alert probability automata shows as below. The green line represents the connection probability below the threshold, the red line is for the missing connection, which doesn't occur in the training automata and the black line shows the connections without alerts.



The figure in the following shows the alert-automata with prefix size 1 and 2, using ALERT_TPYE = RETURN_ONE and CHECKING_TYPE = KEEP_ALL_ENENT.



2.2.4 Server Logging

The server offers two functions for logging.

- log_info() - It is used to log *info* messages
- log_error() - It is used to log *error* messages

We we will have three types of format stored in log file.

1. logging any normal event
   Timesstamp-MessageType-Username-FunctionName-Message
2. logging the event specific data
   Timesstamp- MessageType-Username-FunctionName-ThreadId-CaseId-Activity-Message
3. built-in log format for http requests

There are 4 functions for logging info and logging error with different signatures.

log_info(func_name, message)
log_info(func_name, username, message)
log_info(func_name, username, case_id, activity, message)
log_info(func_name, username, thread_id, case_id, activity, message)

log_error(func_name, message)
log_error(func_name, username, message)

log_error(func_name, username, case_id, activity, message)
log_error(func_name, username, thread_id, case_id, activity, message)

We can change the default values for logging level, time format and filename by manipulating the streaming_event_compliance/__init__.py file for variables LOG_LEVEL and LOG_FORMAT, SERVER_LOG_PATH respectively.

2.2.5 Reconstruction

1. In order to ensure that the global class will be created only once. We reconstruct our code and use singleton to guarantee this.

```
class Singleton(object):
    _instance = None
    def __new__(cls, *args, **kw):
        if not cls._instance:
            cls._instance = super(Singleton, cls).__new__(cls, *args, **kw)
        return cls._instance
```

2. We use *config.ini*, which can provide to the adminstrator, instead of using *config.py*.

3 Testing Project

3.1 Execute time

Testing compliance checking using command: ***pytest -s test/test_flask.py****,* the results of two different event log will show as below: The first one is the same log we used for training, so there is only one alert of type ‘low threshold’, while the second result is from different event log, so all alerts are the type of missing connections.
This report about speed (more than 1000 events per second) is based on running compliance via testing(Request and Checking, doesn’t contain the time of returning results to client). But the real time should be calculated on client side. (We will do it in next stage.)

```
(StreamEC) Jingjings-MacBook-Pro:StreamingEventCompliance jingjinghuo$ pytest -s test/test_flask.py
============================================= test session starts ==============================================
platform darwin -- Python 3.6.7, pytest-4.0.1, py-1.7.0, pluggy-0.8.0
rootdir: /Users/jingjinghuo/Documents/PycharmProjects/StreamingEventCompliance, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, arraydiff-0.2
collected 3 items

test/test_flask.py ..[17:51:43:976382]Results: : OK:22; Alert T:1; Alert M:0
[17:51:43:976419]Path: : /Users/jingjinghuo/Documents/PycharmProjects/StreamingEventCompliance/streaming_event_compli
ance/utils/../../data/Example_EventLogForTraining_Backup.xes
[17:51:43:976434]Events_number: : 23
[17:51:43:976448]Running time: : 0.019336999999999938
[17:51:43:976460]Average speed: : 1189.4295909396533 per second!

[17:51:56:45280]Results: : OK:0; Alert T:0; Alert M:14401
[17:51:56:45317]Path: : /Users/jingjinghuo/Documents/PycharmProjects/StreamingEventCompliance/streaming_event_complia
nce/utils/../../data/A4.xes
[17:51:56:45331]Events_number: : 14401
[17:51:56:45345]Running time: : 11.729000000000001
[17:51:56:45357]Average speed: : 1227.8114076221332 per second!

.

============================================= warnings summary =============================================
/Users/jingjinghuo/anaconda3/envs/StreamEC/lib/python3.6/site-packages/flask_sqlalchemy/__init__.py:794
  /Users/jingjinghuo/anaconda3/envs/StreamEC/lib/python3.6/site-packages/flask_sqlalchemy/__init__.py:794: FSADepreca
tionWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future.
 Set it to True or False to suppress this warning.
    'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '

-- Docs: https://docs.pytest.org/en/latest/warnings.html
================================== 3 passed, 1 warnings in 12.58 seconds ===================================
```

3.2 Unittest

3.3 Comparison for different options

Alert type:

For the case ‘17409’, if we use option RETURN_ALL_ALERTS, then the alerts of type missing will return four times with different window size, while if we use option RETURN_ONE, only the alert with the smallest window size will be return.

1. RETURN_ALL_ALERTS

```
[14:9:35:430682]Alert M :  no such connection in case '174090'
    The connection: W_Complete_Application --> W_Complete_Application
[14:9:35:430716]Alert M :  no such connection in case '174090'
    The connection: App_Pre_Acceptation,W_Complete_Application --> W_Complete_Application,W_Complete_Application
[14:9:35:430791]Alert M :  no such connection in case '174090'
    The connection: App_Incomplete_Submission,App_Pre_Acceptation,W_Complete_Application --> App_Pre_Acceptation,W_Complete_Application,W_Compl
ete_Application
[14:9:35:430908]Alert M :  no such connection in case '174090'
    The connection: App_Fully_Submission,App_Incomplete_Submission,App_Pre_Acceptation,W_Complete_Application --> App_Incomplete_Submission,App
_Pre_Acceptation,W_Complete_Application,W_Complete_Application
```

2. RETURN_ONE(default)

```
[14:21:19:574441]Alert M :  no such connection in case '174111'
    The connection: W_Complete_Application --> W_Handling_Leads
[14:21:19:577666]Alert M :  no such connection in case '174090'
    The connection: W_Complete_Application --> W_Complete_Application
[14:21:19:580968]Alert M :  no such connection in case '174111'
    The connection: W_Handling_Leads --> W_Complete_Application
```

Checking type:

For case ‘173709’, it has 7 events as below, the first alert will occur when we check compliance for the event ‘Err_Pre_Acceptation-complete’, if we use option DELETE_M_EVENT, after we get the alert that no connection from App_Incomplete_Submission-complete to Err_Pre_Acceptation-complete, we delete the Err_Pre_Acceptation-complete, then the second alert will become no connection from App_Incomplete_Submission-complete to W_Complete_Application-complete and so on.

	Activity	Resource	Date	Time	(case) creator	(case) variant	(case) variant-index
1	App_Fully_Submission-complete	112	01.10.2011	09:57:42	Fluxicon Disco	Variant 36	36
2	App_Incomplete_Submission-complete	112	01.10.2011	09:57:43	Fluxicon Disco	Variant 36	36
3	Err_Pre_Acceptation-complete	112	01.10.2011	09:58:27	Fluxicon Disco	Variant 36	36
4	W_Complete_Application-complete	112	01.10.2011	09:58:27	Fluxicon Disco	Variant 36	36
5	W_Complete_Application-complete	10912	01.10.2011	10:26:42	Fluxicon Disco	Variant 36	36
6	W_Complete_Application-complete	10912	01.10.2011	10:27:07	Fluxicon Disco	Variant 36	36
7	W_Complete_Application-complete	10912	01.10.2011	11:40:40	Fluxicon Disco	Variant 36	36

1. KEEP_ALL_EVENTS(default)

```
[14:37:46:914673]Alert M :  no such connection in case '173709'
    The connection: App_Incomplete_Submission --> Err_Pre_Acceptation
    The expected connection:
         App_Incomplete_Submission --> App_Rejection :  0.275144
         App_Incomplete_Submission --> App_Pre_Acceptation :  0.353482
         App_Incomplete_Submission --> W_Handling_Leads :  0.36619
         App_Incomplete_Submission --> W_Fraud_Detection :  0.00518351
[14:37:46:918566]Alert M :  no such connection in case '173709'
    The connection: Err_Pre_Acceptation --> W_Complete_Application
[14:37:47:161989]Alert M :  no such connection in case '173709'
    The connection: Err_Pre_Acceptation,W_Complete_Application --> W_Complete_Application,W_Complete_Application
[14:37:47:573742]Alert M :  no such connection in case '173709'
    The connection: Err_Pre_Acceptation,W_Complete_Application,W_Complete_Application --> W_Complete_Application,W_Complete_Application,W_Complete_Application
[14:37:48:193552]Alert M :  no such connection in case '173709'
    The connection: Err_Pre_Acceptation,W_Complete_Application,W_Complete_Application,W_Complete_Application --> W_Complete_Application,W_Complete_Application
,W_Complete_Application,W_Complete_Application
```

2. DELETE_M_EVENT

```
[14:34:46:870922]Alert M :  no such connection in case '173709'
    The connection: App_Incomplete_Submission --> Err_Pre_Acceptation
    The expected connection:
         App_Incomplete_Submission --> App_Rejection :  0.275144
         App_Incomplete_Submission --> App_Pre_Acceptation :  0.353482
         App_Incomplete_Submission --> W_Handling_Leads :  0.36619
         App_Incomplete_Submission --> W_Fraud_Detection :  0.00518351
[14:34:46:874841]Alert M :  no such connection in case '173709'
    The connection: App_Incomplete_Submission --> W_Complete_Application
    The expected connection:
         App_Incomplete_Submission --> App_Rejection :  0.275144
         App_Incomplete_Submission --> App_Pre_Acceptation :  0.353482
         App_Incomplete_Submission --> W_Handling_Leads :  0.36619
         App_Incomplete_Submission --> W_Fraud_Detection :  0.00518351
[14:34:46:878291]Alert M :  no such connection in case '173709'
    The connection: App_Incomplete_Submission --> W_Complete_Application
    The expected connection:
         App_Incomplete_Submission --> App_Rejection :  0.275144
         App_Incomplete_Submission --> App_Pre_Acceptation :  0.353482
         App_Incomplete_Submission --> W_Handling_Leads :  0.36619
         App_Incomplete_Submission --> W_Fraud_Detection :  0.00518351
[14:34:46:881782]Alert M :  no such connection in case '173709'
    The connection: App_Incomplete_Submission --> W_Complete_Application
    The expected connection:
         App_Incomplete_Submission --> App_Rejection :  0.275144
         App_Incomplete_Submission --> App_Pre_Acceptation :  0.353482
         App_Incomplete_Submission --> W_Handling_Leads :  0.36619
         App_Incomplete_Submission --> W_Fraud_Detection :  0.00518351
[14:34:46:885171]Alert M :  no such connection in case '173709'
    The connection: App_Incomplete_Submission --> W_Complete_Application
    The expected connection:
         App_Incomplete_Submission --> App_Rejection :  0.275144
         App_Incomplete_Submission --> App_Pre_Acceptation :  0.353482
         App_Incomplete_Submission --> W_Handling_Leads :  0.36619
         App_Incomplete_Submission --> W_Fraud_Detection :  0.00518351
```

Window Size: the default value is [1,2,3,4], and one can change it into whatever he wants to have, like [1,2,3,4,5,6] or [2,4,6] etc.[1, 2, 3]

If the user give the window size [3, 4, 6], then as we can see in the following picture, the checking begins with window size 3, and then do 4 and 5.

```
[16:12:51:756706]DEFAULT : WINDOW_SIZE: [1, 2, 3, 4] CHECKING_TYPE: KEEP_ALL_EVENTS  ALERT_TYPE: RETURN_ONE
[16:12:51:757894]PATH : /Users/jingjinghuo/Documents/PycharmProjects/StreamingEventCompliance/streaming_event_compliance/../data/A4.xes
[16:12:51:757930]USER-DEFINED : WINDOW_SIZE: [3, 4, 6]  MAXIMUN_WINDOW_SIZE: 6  CHECKING_TYPE: KEEP_ALL_EVENTS  ALERT_TYPE: RETURN_ALL_ALERTS
```

```
(StreamEC) 080-139:StreamingEventCompliance jingjinghuo$ python client/client.py z219 client/Example_EventLogForTesting.xes
There are two services:
       Press 1, if you want to do the compliance checking
       Press 2, if you want to show the deviation pdf
       Press 3, if you want to exit
[16:12:59:719676]Note: : you can interrupt with CTR_C, once you start to do the compliance checking
1
[16:13:0:476201][ INFO  ] : ---------------start to do compliance checking, please wait-------------------
[16:13:0:547927]Alert M :  no such start node' App_Fully_Submission,App_Incomplete_Submission,Err_Pre_Acceptation 'in case ' 173709'
    The expected start node:
        ' App_Fully_Submission,App_Incomplete_Submission,App_Pre_Acceptation ' with probability:  0.623441
        ' App_Fully_Submission,App_Incomplete_Submission,W_Handling_Leads ' with probability:  0.337905
        ' App_Fully_Submission,App_Incomplete_Submission,W_Fraud_Detection ' with probability:  0.0386534
[16:13:0:598899]Alert M :  no such connection in case '173709'
    The connection: App_Fully_Submission,App_Incomplete_Submission,Err_Pre_Acceptation --> App_Incomplete_Submission,Err_Pre_Acceptation,W_Complete_Applicatio
n
[16:13:0:598942]Alert M :  no such start node' App_Fully_Submission,App_Incomplete_Submission,Err_Pre_Acceptation,W_Complete_Application 'in case ' 173709'
    The expected start node:
        ' App_Fully_Submission,App_Incomplete_Submission,App_Pre_Acceptation,W_Complete_Application ' with probability:  0.623441
        ' App_Fully_Submission,App_Incomplete_Submission,W_Handling_Leads,W_Handling_Leads ' with probability:  0.337905
        ' App_Fully_Submission,App_Incomplete_Submission,W_Fraud_Detection,W_Fraud_Detection ' with probability:  0.0386534
[16:13:0:602815]Alert M :  no such connection in case '173709'
    The connection: App_Incomplete_Submission,Err_Pre_Acceptation,W_Complete_Application --> Err_Pre_Acceptation,W_Complete_Application,W_Complete_Application
[16:13:0:602856]Alert M :  no such connection in case '173709'
    The connection: App_Fully_Submission,App_Incomplete_Submission,Err_Pre_Acceptation,W_Complete_Application --> App_Incomplete_Submission,Err_Pre_Acceptatio
n,W_Complete_Application,W_Complete_Application
[16:13:0:663951]Alert M :  no such connection in case '173709'
    The connection: Err_Pre_Acceptation,W_Complete_Application,W_Complete_Application --> W_Complete_Application,W_Complete_Application,W_Complete_Application
[16:13:0:663992]Alert M :  no such connection in case '173709'
    The connection: App_Incomplete_Submission,Err_Pre_Acceptation,W_Complete_Application,W_Complete_Application --> Err_Pre_Acceptation,W_Complete_Application
,W_Complete_Application,W_Complete_Application
[16:13:0:664015]Alert M :  no such start node' App_Fully_Submission,App_Incomplete_Submission,Err_Pre_Acceptation,W_Complete_Application,W_Complete_Applicatio
n,W_Complete_Application 'in case ' 173709'
    The expected start node:
        ' App_Fully_Submission,App_Incomplete_Submission,App_Pre_Acceptation,W_Complete_Application,W_Complete_Application,W_Complete_Application ' with proba
bility:  0.49
        ' App_Fully_Submission,App_Incomplete_Submission,W_Handling_Leads,W_Handling_Leads,App_Pre_Acceptation,W_Complete_Application ' with probability:  0.2
7625
        ' App_Fully_Submission,App_Incomplete_Submission,W_Handling_Leads,W_Handling_Leads,W_Handling_Leads,W_Handling_Leads ' with probability:  0.06125
        ' App_Fully_Submission,App_Incomplete_Submission,App_Pre_Acceptation,W_Complete_Application,W_Complete_Application,App_Acceptation ' with probability:
  0.13375
        ' App_Fully_Submission,App_Incomplete_Submission,W_Fraud_Detection,W_Fraud_Detection,W_Fraud_Detection,W_Fraud_Detection ' with probability:  0.03875
```

4 Remaining Tasks

PDF-Impovement - more beautiful
Correctness
Code Comments
Deployment

5 Phase Review

Zheqi Lyu: In this phase, we try to finish the majority of the project and make our project more flexible and scalable. It became the difficult part. Because a small change may occur many overwriting. It may still have some small bugs we didn't find. And we will keep on the correctness in the next phase. I hope we can provide a project with clear skeleton and interface at the end.
Sabya Shaikh: We have almost finished the project. We yet have to improvise minor parts. As I was out of town I completed most of my tasks before leaving aachen. We kept in touch with messages and co-ordinated well enough for this sprint. There were alot of changes in the code to improvise it. I tried to give the best of my time for documentation too. Thanks to my team mates for co-ordinating well with me even when I was away from town.
Jingjing Huo: At the beginning, I thought that we don’t have much work to do, but as we did the coding, problems come one by one, like give different options for compliance checking, restructure the code and so on. And I think we have done the most thing of the project, in the next phase, we can do something like improving the robustness of our code.