REBS_Assignment_1

Group: Assignment 2
NJX155: Lars KL
hck787: Hu Guo
tnd179: Xingrong Zong
Josiah Grønhøj

16th of December 2021

Part 1

1) Fill out application should always be the first event of the case.
This is a condition that says that filling out application must be the first activity before anything else in the process. No other activity must be executed before *Fill out application* according to this pattern.
2) Reject should always eventually be followed by *Applicant informed* and *Change phase to Abort*
This is a response that says that whenever a *Reject* occurs then this must eventually be followed by the following two activities: *Applicant informed* and *Change phase to Abort*. Practically this means that whenever the system rejects an application then at some point the applicant must be informed about it and the procedural phase must be aborted.
3) *First payment* should only occur once.
This is an exclusion. Whenever the event *First payment* has been executed the event will delete itself from the process and can thus not be executed again while the process is on-going. Practically this means that a payment must not be able to happen more than once in the process.
4) *Lawyer Review* and *Architect Review* should never occur together.
This is also an exclusion, but in this case there are two exclusions between two events meaning that when one event is executed, it will exclude the other event from being executed and vice versa. Practically this means that during the execution of the process we cannot have both a lawyer review and a architect review occuring in the same process. We create the graph which excludes both occuring together.

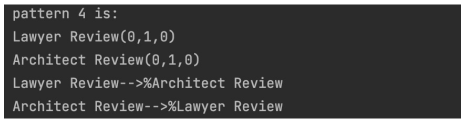
Part 2: DCR Graph Conformance Checker

Implementation

The whole implementation consists of following 5 parts:
1. Class **ParseLog**: Parse the logs. The result of this part is a HashMap⟨String, List⟨String⟩⟩ map, where the key is the log's id and the value is a list of events. There are 594 logs in total.
2. Class **PatternGenerator**: For each DCR in question 1, we generate a pattern for it, represented as a program code. There are 2 types of patterns: one is to set up the initial marks for each event such as ”Change phase to Abort(0,1,0)”. The other is to describe the relationship between events such as ”Reject *—> Applicant informed”.
3. Implementation of **DCRGraph** Class: the implementation refers to the code in the slides. A DCRGraph instance have events, the relationships between events and marking states of events. The code contains all 5 relationships: condition, milestone, response, exclude and include, but the patterns only contains condition, response and exclude. Function **enable** in this class receives a marking states and an event, and judge if this event is enabled under these states and the relationship. Function **execute** returns the marking states after executing an event.
4. Class **ParseDCR**: The generated patterns are represented as some ”code” and we should transform these codes to the information in a specific DCRGraph instance.
5. For each pattern and each log, we first initiate a DCRGraph instance for it with the initial mark states and the given relationship, using the method in **ParseDCR**. Then for each log, we use following way to judge if it is accepted by the pattern: we execute the events of this log in order, and during this period, if one of the event is not enabled, this log will not be accepted. Finally, if the final DCRGraph is an accepted state, it means the log is accepted.

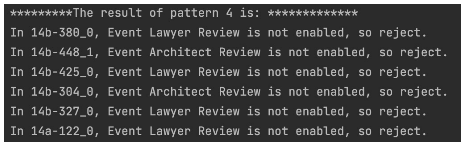
Running

Take pattern 4 as the example, the output to the terminal includes the pattern, the total amount of logs and how many logs are accepted by the pattern. Also if one log is rejected by one pattern, the reason is also printed. The figures below show the running outputs for pattern 4. The other 3 patterns' running snapshots can be found when running the JAVA code.



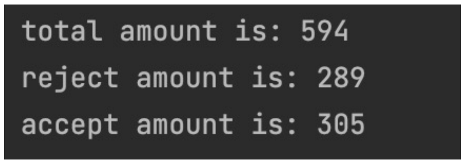
```
pattern 4 is:
Lawyer Review(0,1,0)
Architect Review(0,1,0)
Lawyer Review-->%Architect Review
Architect Review-->%Lawyer Review
```

Figure 1: Description of Pattern 4



```
*********The result of pattern 4 is: *************
In 14b-380_0, Event Lawyer Review is not enabled, so reject.
In 14b-448_1, Event Architect Review is not enabled, so reject.
In 14b-425_0, Event Lawyer Review is not enabled, so reject.
In 14b-304_0, Event Architect Review is not enabled, so reject.
In 14b-327_0, Event Lawyer Review is not enabled, so reject.
In 14a-122_0, Event Lawyer Review is not enabled, so reject.
```

Figure 2: Some Reject Information of Pattern 4



```
total amount is: 594
reject amount is: 289
accept amount is: 305
```

Figure 3: Result for Pattern 4

Results

The results is showed below. For pattern 3, we use the "exactly once" rather than "at most once". If the semantic is "at most once", all the logs are accepted by this pattern.

	total	accept	reject
pattern 1	594	594	0
pattern 2	594	594	0
pattern 3	594	576	18
pattern 4	594	305	289