Project 6 – Report
Research on AGORA

By Abhishek Manyam, Anushka Jain, Rishitha Pokalkar

1. Research Question

RQ1 - Problem Statement: The study investigates the extent to which modifications in API specifications, including variable types, impact AGORA's ability to generate accurate and relevant test oracles for REST APIs.

RQ2 - Problem Statement: This study examines how changes in black-box test case generators, particularly transitioning from RESTest to alternative tools like RESTler and RESTTestGen, affect the patterns and quality of invariants detected in REST API testing.

2. Motivation

Motivation for RQ1: Understanding this issue is crucial to aid developers in strategically modifying their APIs for improved test oracle generation. As the dynamic nature of REST API development leads to frequent adjustments in structures, such as paths and data types, developers need insights into whether such changes compromise the integrity and applicability of generated test oracles. Identifying patterns that enhance test oracle generation will empower developers to balance API evolution with maintaining high-quality, reliable automated testing processes.

Motivation for RQ2: The research aims to critically evaluate the influence of test case generator selection on detecting invariants, which are essential for identifying consistent behaviors and anomalies in REST APIs. By assessing the impact of transitioning between tools, developers and testers can gain valuable insights into selecting the most appropriate tool for their testing needs. Ultimately, this will contribute to the development of more reliable and high-quality APIs.

3. Relevance

Research Question 1:
The problem revolves around understanding how modifications in API specifications affect the ability of automated testing tools, specifically AGORA, to generate accurate test oracles. This aligns with previous research focused on automated testing in the context of dynamic API environments. While existing studies have explored various aspects of API testing, including test case generation and validation, the specific impact of API specification modifications on test oracle generation may not have been extensively addressed. By investigating this aspect, the research aims to contribute valuable insights into maintaining the effectiveness of automated testing processes amidst evolving APIs.

Research Question 2:
The problem addressed here concerns the influence of different black-box test case generators, such as RESTest, RESTler, and RESTTestGen, on the detection of invariants in REST API testing. This aligns with prior research on test case generation techniques and tools for API testing. While existing literature may have evaluated individual test case generators or compared their performance in specific contexts, the direct comparison of these tools concerning invariant detection and quality assessment could be relatively unexplored. By examining this aspect, the research seeks to provide developers and testers with actionable insights into selecting appropriate testing tools, thereby contributing to the advancement of reliable and high-quality API development practices.

4. Methodology

Our solution strategy began by addressing two primary research questions. Initially, we tackled the issue by altering data types within the OAS (OpenAPI Specification) specifications, specifically transitioning from integers to strings. This change was deemed necessary as converting data types from strings to integers or booleans to integers would lead to errors, as characters cannot be seamlessly passed with integer input types into the datatrace file.

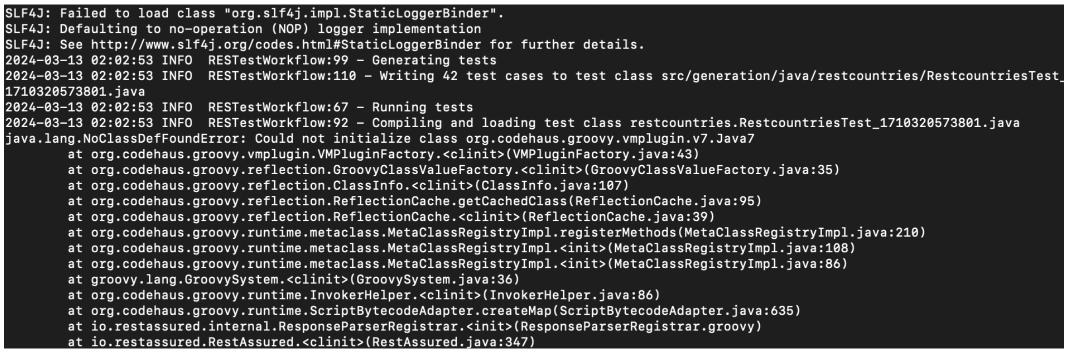
Addressing Research Question 1:

Steps Undertaken:

1) We selected APIs utilized in the referenced paper, including Spotify, Amadeus, Github, and Marvel, and identified specific parameters to address.
   a) Spotify (offset, limit, tracknumber)
   b) Amadeus (adults, room quantity)
   c) Github (Team-id)
   d) Marvel (comic_id, available)
2) Utilizing the existing OAS specifications and test dataset from the paper, we fed them into Beet to generate declaration and datatrace files (referred to as olddeclaration and olddetrace).
3) Invariants_old were then generated using the files from the previous step.
4) Subsequently, we modified the OAS specifications for the forementioned APIs to reflect the necessary changes.
5) Employing the updated OAS specifications, we generated new declaration and datatrace files (newdeclaration and newdetrace).
6) Finally, new invariants were derived using the files from step 5.

Addressing Research Question 2:

Initially, we assumed that Restest would operate seamlessly. However, we encountered a setback due to a Groovy dependency issue in Maven, persisting across various versions of Restest (1.2.0, 1.3.0, 1.4.0) and across different platforms (MacOS and Windows) using Jbr 17 (Jetbrains), JDK 17, and OpenJDK 21.



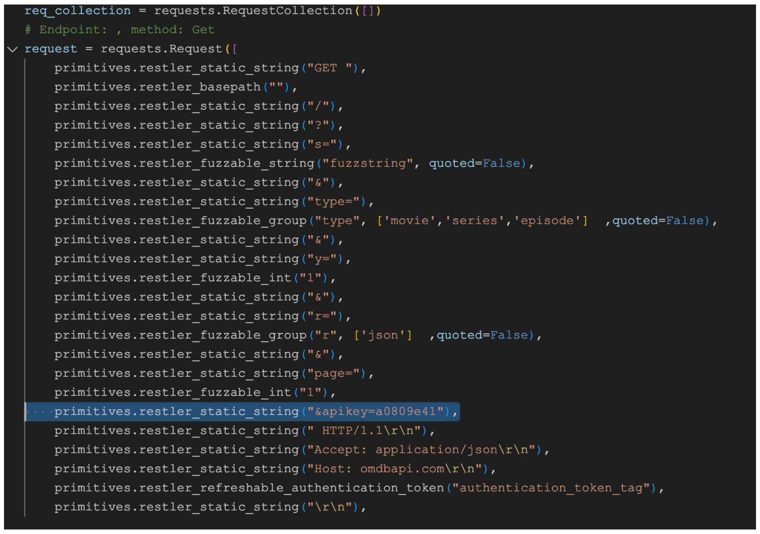
```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
2024-03-13 02:02:53 INFO  RESTestWorkflow:99 - Generating tests
2024-03-13 02:02:53 INFO  RESTestWorkflow:110 - Writing 42 test cases to test class src/generation/java/restcountries/RestcountriesTest_
1710320573801.java
2024-03-13 02:02:53 INFO  RESTestWorkflow:67 - Running tests
2024-03-13 02:02:53 INFO  RESTestWorkflow:92 - Compiling and loading test class restcountries.RestcountriesTest_1710320573801.java
java.lang.NoClassDefFoundError: Could not initialize class org.codehaus.groovy.vmplugin.v7.Java7
        at org.codehaus.groovy.vmplugin.VMPluginFactory.<clinit>(VMPluginFactory.java:43)
        at org.codehaus.groovy.reflection.GroovyClassValueFactory.<clinit>(GroovyClassValueFactory.java:35)
        at org.codehaus.groovy.reflection.ClassInfo.<clinit>(ClassInfo.java:107)
        at org.codehaus.groovy.reflection.ReflectionCache.getCachedClass(ReflectionCache.java:95)
        at org.codehaus.groovy.reflection.ReflectionCache.<clinit>(ReflectionCache.java:39)
        at org.codehaus.groovy.runtime.metaclass.MetaClassRegistryImpl.registerMethods(MetaClassRegistryImpl.java:210)
        at org.codehaus.groovy.runtime.metaclass.MetaClassRegistryImpl.<init>(MetaClassRegistryImpl.java:108)
        at org.codehaus.groovy.runtime.metaclass.MetaClassRegistryImpl.<init>(MetaClassRegistryImpl.java:86)
        at groovy.lang.GroovySystem.<clinit>(GroovySystem.java:36)
        at org.codehaus.groovy.runtime.InvokerHelper.<clinit>(InvokerHelper.java:86)
        at org.codehaus.groovy.runtime.ScriptBytecodeAdapter.createMap(ScriptBytecodeAdapter.java:635)
        at io.restassured.internal.ResponseParserRegistrar.<init>(ResponseParserRegistrar.groovy)
        at io.restassured.RestAssured.<clinit>(RestAssured.java:347)
```

Figure 1 : Failed ResTest compilation due to Groovy dependancy.

To overcome this obstacle, we opted to focus solely on APIs for which Restest outputs were already available. We then utilized Restler and RestestGen to generate test cases. However, this approach presented challenges as these tools were primarily designed for API testing rather than test case generation.

Challenges with Restler and RestestGen:

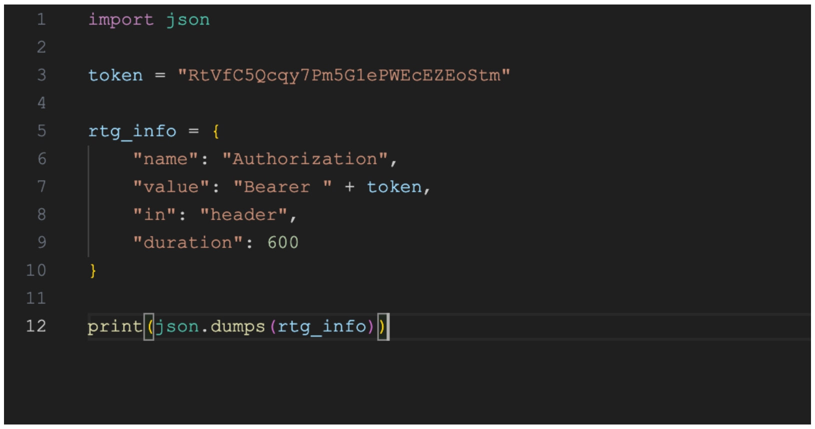
- **Restler:** All APIs mentioned in the paper employed authentication techniques. We obtained API keys for Marvel, Amadeus, and OMDB. While Marvel and Amadeus required complex authentication methods, OMDB simply required passing the API key as a query variable. Adapting Restler's grammar to accommodate query-based authentication proved necessary, although the functional correctness of this approach remains unverified.



```
req_collection = requests.RequestCollection([])
# Endpoint: , method: Get
request = requests.Request([
    primitives.restler_static_string("GET "),
    primitives.restler_basepath(""),
    primitives.restler_static_string("/"),
    primitives.restler_static_string("?"),
    primitives.restler_static_string("s="),
    primitives.restler_fuzzable_string("fuzzstring", quoted=False),
    primitives.restler_static_string("&"),
    primitives.restler_static_string("type="),
    primitives.restler_fuzzable_group("type", ['movie','series','episode']  ,quoted=False),
    primitives.restler_static_string("&"),
    primitives.restler_static_string("y="),
    primitives.restler_fuzzable_int("1"),
    primitives.restler_static_string("&"),
    primitives.restler_static_string("r="),
    primitives.restler_fuzzable_group("r", ['json']  ,quoted=False),
    primitives.restler_static_string("&"),
    primitives.restler_static_string("page="),
    primitives.restler_fuzzable_int("1"),
    primitives.restler_static_string("&apikey=a0809e41"),
    primitives.restler_static_string(" HTTP/1.1\r\n"),
    primitives.restler_static_string("Accept: application/json\r\n"),
    primitives.restler_static_string("Host: omdbapi.com\r\n"),
    primitives.restler_refreshable_authentication_token("authentication_token_tag"),
    primitives.restler_static_string("\r\n"),
```

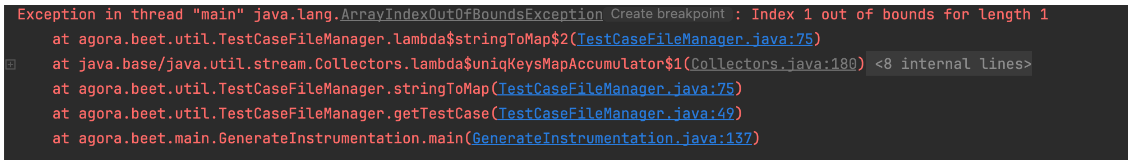
Figure 2 : Added api key to grammar.py file

- **RestTestGen:** Producing tests with RestTestGen was comparatively straightforward. However, converting the generated JSON files into a CSV format readable by Beet posed a challenge. We specifically encountered difficulties in providing Beet with requisite details such as pathParameters, formParameters, and bodyParameters, which were not seamlessly generated by Restler and RestTestGen.



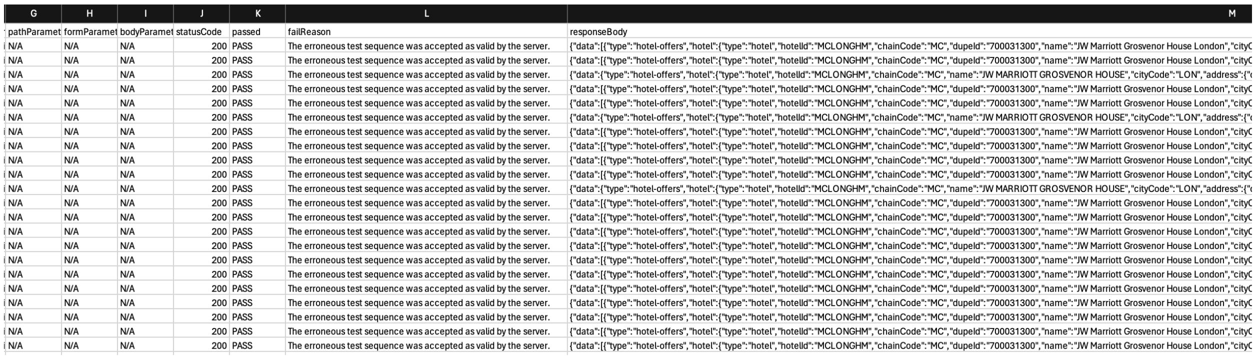
```python
import json

token = "RtVfC5Qcqy7Pm5G1ePWEcEZEoStm"

rtg_info = {
    "name": "Authorization",
    "value": "Bearer " + token,
    "in": "header",
    "duration": 600
}

print(json.dumps(rtg_info))
```

Figure 3 : Authentication script used by RestestGen sent using header.



```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException Create breakpoint : Index 1 out of bounds for length 1
    at agora.beet.util.TestCaseFileManager.lambda$stringToMap$2(TestCaseFileManager.java:75)
    at java.base/java.util.stream.Collectors.lambda$uniqKeysMapAccumulator$1(Collectors.java:180) <8 internal lines>
    at agora.beet.util.TestCaseFileManager.stringToMap(TestCaseFileManager.java:75)
    at agora.beet.util.TestCaseFileManager.getTestCase(TestCaseFileManager.java:49)
    at agora.beet.main.GenerateInstrumentation.main(GenerateInstrumentation.java:137)
```

Figure 4 : Error running beet on the new CSV



G	H	I	J	K	L	M
pathParamet	formParamet	bodyParamet	statusCode	passed	failReason	responseBody
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","name":"JW MARRIOTT GROSVENOR HOUSE","cityCode":"LON","address":{"
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","name":"JW MARRIOTT GROSVENOR HOUSE","cityCode":"LON","address":{"
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","name":"JW MARRIOTT GROSVENOR HOUSE","cityCode":"LON","address":{"
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC
N/A	N/A	N/A	200	PASS	The erroneous test sequence was accepted as valid by the server.	{"data":[{"type":"hotel-offers","hotel":{"type":"hotel","hotelId":"MCLONGHM","chainCode":"MC","dupeId":"700031300","name":"JW Marriott Grosvenor House London","cityC

Figure 5 : Json converted into csv

Conclusion:

Despite efforts to integrate Restler and RestestGen into the testing process, unaddressed challenges remain, particularly concerning data formatting and parameter handling required by Beet. Additional research and potential modifications to these tools may be necessary to effectively integrate them into AGORA for comprehensive testing purposes.

5. Results

Our investigation into Research Question 1 began with two main aspects:

a) Number of Invariants Produced:

We started by comparing the count of invariants before and after altering the data types across the four APIs we tested (Spotify, Amadeus, Github, and Marvel). After examining the data, we noticed that for three out of the four APIs, there was a reduction in the number of invariants following the data type modification. The change is visually illustrated in the graph provided below:

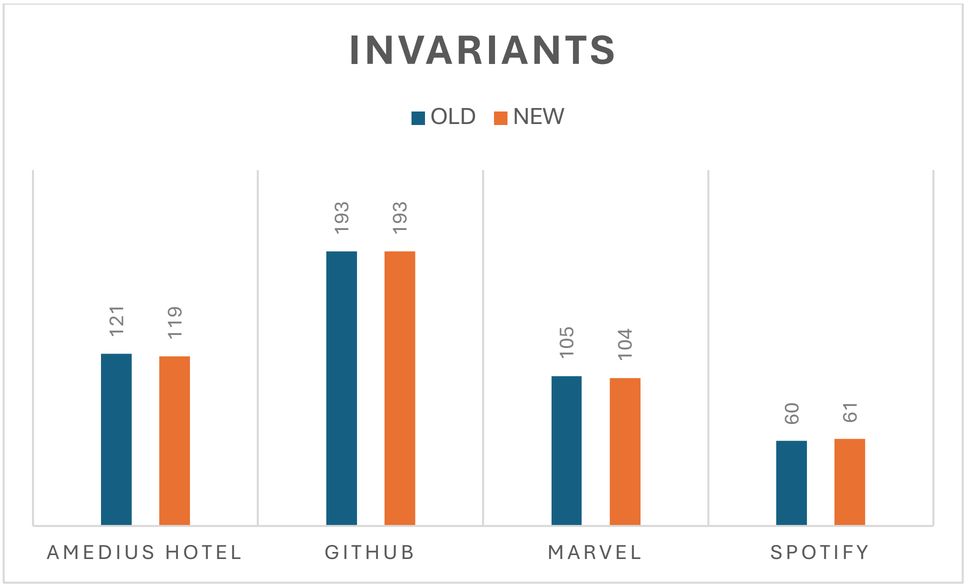
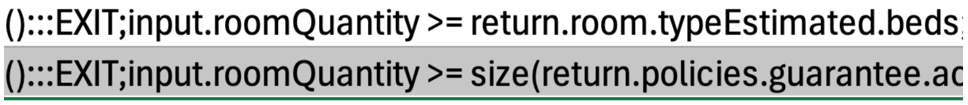


Figure 6 : Count of invariant in Old and new csv's

b) Difference in the Invariants: To delve deeper into the analysis, we conducted a comparison of the invariants themselves before and after the data type change. This involved using Excel's VLOOKUP operation to identify any missing or newly introduced invariants.

Findings Specific to Each API:

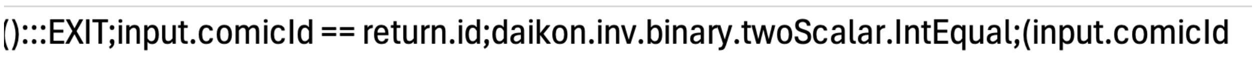
Amadeus Hotel API: After the data type alteration, we observed that two invariants were absent in the new CSV compared to the old one. The absence of these invariants is highlighted in the figure below:



```
():::EXIT;input.roomQuantity >= return.room.typeEstimated.beds
():::EXIT;input.roomQuantity >= size(return.policies.guarantee.ac
```

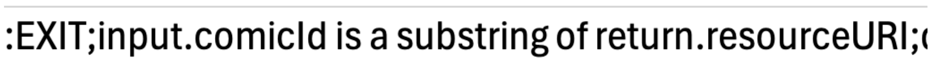
Marvel API: Notably, we found a change in the invariant for 'comicID', where the old invariant was replaced with a new one. Additionally, there was a loss of an invariant that wasn't directly linked to the variable subjected to the data type change.

Old:



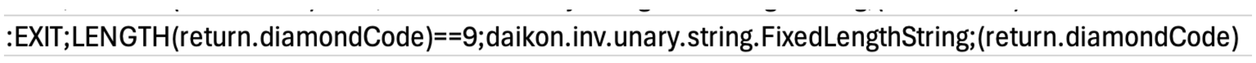
```
():::EXIT;input.comicId == return.id;daikon.inv.binary.twoScalar.IntEqual;(input.comicId
```

New:



```
:EXIT;input.comicId is a substring of return.resourceURI;
```

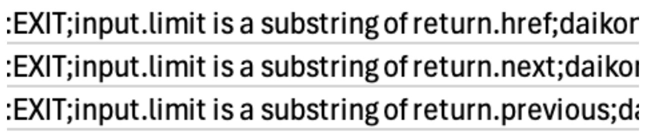
Lost:



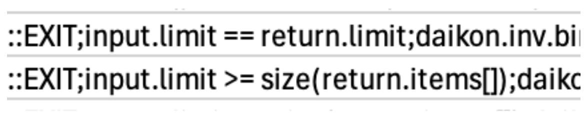
```
:EXIT;LENGTH(return.diamondCode)==9;daikon.inv.unary.string.FixedLengthString;(return.diamondCode)
```

Spotify API: Upon analysis, it was evident that three new invariants emerged in the new CSV, replacing two old ones from the previous CSV. The specific invariants are showcased below:

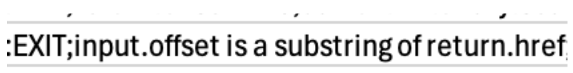
New:


```
:EXIT;input.limit is a substring of return.href;daikor
:EXIT;input.limit is a substring of return.next;daiko
:EXIT;input.limit is a substring of return.previous;d
```

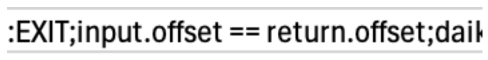
Old:


```
::EXIT;input.limit == return.limit;daikon.inv.bi
::EXIT;input.limit >= size(return.items[]);daik
```

New2:


```
:EXIT;input.offset is a substring of return.href
```

Old2:


```
:EXIT;input.offset == return.offset;dail
```

Upon reviewing these results, we revisited the paper's statement which states that “The largest portion of false positives (75.8%) were found in the arithmetic comparison category”. By changing the datatype from int to string we can see there is a clear decrease in the usage of arithmetic comparison without the sacrifice of number of invariants. This observation suggests that altering data types can effectively reduce arithmetic comparisons, thereby indirectly mitigating false positives.

Github- https://github.com/medhyaj/AGORA_project

Trello- https://trello.com/b/jhzWxNJB/project-group6