Test Report for ECA Rules for Ampersand

Yuriy Toporovskyy (toporoy)
Yash Sapra (sapray)
Jaeden Guo (guoy34)

March 25th, 2016

Table 1: Revision History

Author	**Date**	**Comments**
Yash Sapra	24 / 03 / 2016	Initial draft
Yash Sapra	24 / 03 / 2016	Performance Testing

Contents

1 Introduction

1.1 Description

This document details the test results of the EFA project. This document uses the test description mentioned in the test plan. EFA, as well as the core Ampersand system, is currently in active development where changes occur frequently. For this reason few tests could not performed. A second phase of testing will be performed once the EFA project is integrated into the core Ampersand. The original test plan is available in the github repository and is being actively revised in team meetings. Changes to test plan will follow soon.

1.2 Scope

The purpose of this document is to outline the implementation details of the EFA project described in the Problem Statement. EFA is responsible for generating SQL Statements from ECA rules that will be used to fixed any violated invariants in the Ampersand prototype. The document will serve as a referral document for future software Testing and integration of EFA in the Ampersand project.

1.3 Test Cases

For the purpose of testing, the EFA team uses the .adl files from the ampersand-models repository. This repository contains various input files for the Ampersand Core project. Any files that compiles and runs with the core Ampersand software should also run accordingly with the EFA project.

2 Definitions

Sentinel

A test server accessible through the Ampersand website which executes a set of randomly generated tests on Ampersand on a daily basis.

ECA Rule

Event-Condition-Action Rule. A rule which describes how to handle a constraint violation in a database. The syntax of ECA rules is as follows:

```
ECArule ::= 'On' ('Ins' | 'Del')
            '(' RExpr ',' RAtom ')'
            'Do' PAclause
```

3 Non-Functional Testing

3.1 Usability

From a usability perspective EFA project integrates seamlessly into the current version of core Ampersand. User can use –help flag to view different options they've while generating a prototype. The ‘’- -print-eca-info" flag prints the generated SQL for each ECA rule in the console. This can be useful from a development perspective in future. The Developers and Maintainers of Ampersand can use this flag to evaluate the underlying SQL accompanying each ECA rule described in the .adl file.

This test follows with the test case T11 and completed the functional requirement that the EFA project has to produce annotated code (SQL).

3.2 Performance Testing

The performance test refers to the T10 test case of the EFA project test plan. The EFA team planned to perform a degradation test to performance degradation if any. All the files were compiled with the latest version of core Ampersand and then with the EFA. The results are documented in this section.

No.	**Input File**	**Run-Time Without EFA project**	**Run-Time With EFA project**
1	ProjectAdmin.adl	5.85	7.63
2	Delivery.adl	5.33	6.01
3	Try1.adl	6.16	6.93
4	Try2.adl	5.95	6.45
5	Try3.adl	6.28	7.01
6	Try4.adl	6.78	7.44
7	Try5.adl	6.13	7.1
8	Try6.adl	6.16	7.65
9	Try7.adl	6.98	8.01
10	Try8.adl	7.5	8.65

11	Try9.adl	7.2	8.22
12	Try10.adl	6.33	7.88
13	Try11.adl	6.47	7.57
14	Try12.adl	7.88	8.68
15	Try13.adl	7.56	8.92
16	Try14.adl	7.11	8.75
17	Try15.adl	7.13	9.01
18	Try16.adl	6.15	8.01
19	Try17.adl	6.39	7.66
20	Try18.adl	6.04	7.32
21	Try19.adl	6	6.9
22	Try20.adl	5.62	6.81

After measuring the performance of the current version of Ampersand compared to the EFA project we found out that there is a overhead cost of generating SQL statements from the ECA rules. The average overhead time of running EFA project is 1.16 sec.

Calculate using the formula :

$$OverheadTime(s) = \frac{(\sum RunTimewithEFA - RunTimewithoutEFA)}{No.ofTestCases} \quad (1)$$

Figure1 and Figure2 shows a comparison of running time for all the test cases. The overhead cost of integrating EFA into Ampersand will add roughly about 1 second to the time it takes to generate a prototype. However the overall running time is still under 9 seconds for all the test cases so the waiting time for the end user is still very small compared to cost and time required to create an information system otherwise.

3.3 Robustness

The language dependency of using Haskell for this project allows the Developers to pattern match against all possible inputs. The Project was tested using the '' --Wall" flag to turn on all the warning options in Haskell. This allowed the team to pattern match against all possible inputs, this way the project does not rely on the test cases reachable through the Ampersand test input files.

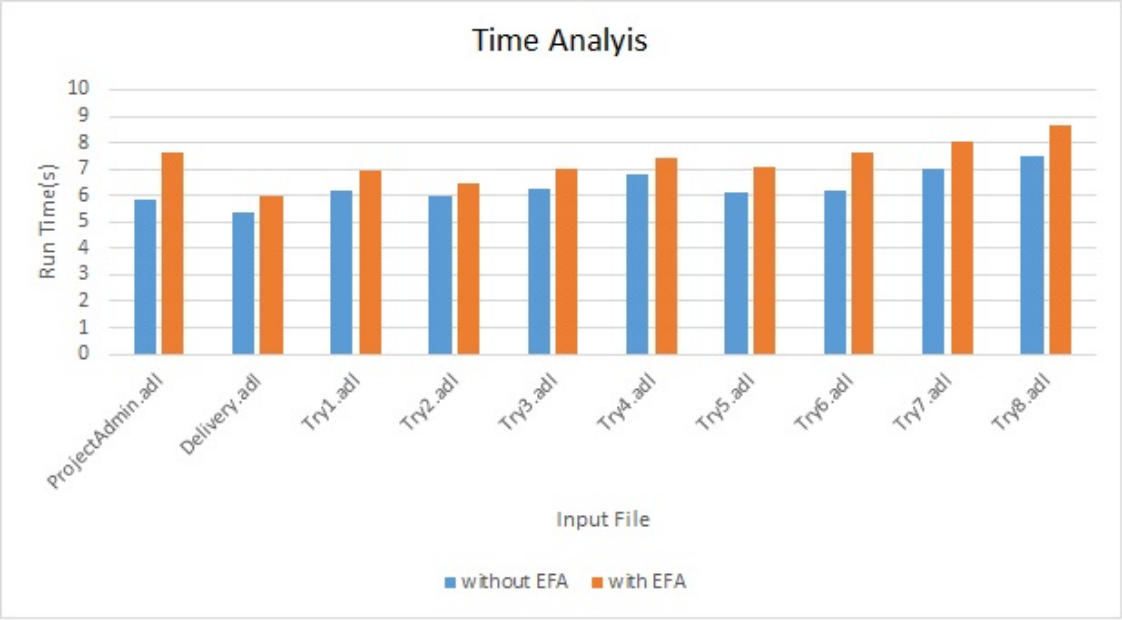

Figure 1: Run Time chart for test case 1 to 10.

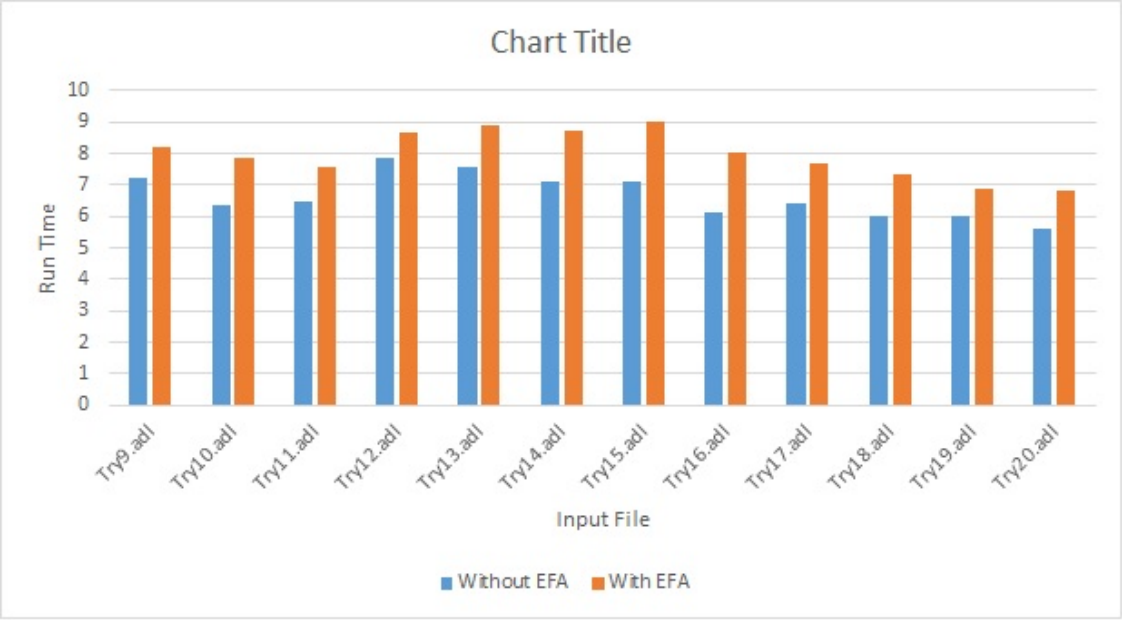

Figure 2: Run Time chart for test case 11 to 22.

4 System Tests

In this section we document the result of parsing ADL files through the EFA project.

4.1 Ampersand generates ASQL

No.	Test Case	Initial State	Input	Expected Output	Actual Output	Result
1	Ampersand generates ASQL	Installed EFA Ampersand	ProjectAdmin.adl	Annotated SQL	As Expected	PASS
2	Ampersand generates ASQL	Installed EFA Ampersand	Delivery.adl	Annotated SQL	As Expected	PASS
3	Ampersand generates ASQL	Installed EFA Ampersand	Case.adl	Annotated SQL	As Expected	PASS

4.2 ASQL is valid

4.3 EFA System Compatibility

No.	Test Case	Initial State	Input	Expected Output	Actual Output	Result
1	System Compatibility	Installed EFA Ampersand	ProjectAdmin.adl	No exception during generation of prototype	As Expected	PASS
2	System Compatibility	Installed EFA Ampersand	Delivery.adl	No exception during generation of prototype	As Expected	PASS
3	System Compatibility	Installed EFA Ampersand	Case.adl	No exception during generation of prototype	As Expected	PASS

4.4 EFA is a pure function

Since all functions written in Haskell are pure, and the Haskell type checker accepts our program hence the test is passed.

4.5 EFA gives appropiate feedback

This feature will be implemented on the front-end after integration into the core Ampersand project. When the prototype is run, and a violation occurs, the resulting output will look like :

```
======= Violation log entry <...>
=== ECA rule fired: <...>
=== Delta: <...>
=== Original rule: cast;instantiates |- qualifies;comprises~
Violation occurred because rule "who's cast in roles" was not
 satisfied. This is because "an Actor may appear in a
 Performance of the Play only if the Actor is skilled for a
 Role that the Play comprises"
```

4.6 EFA code walk-through

With reference to T9 test in the test report (see page 19 of the test plan). EFA team will be doing a code walk-through with the product owners. This walk-through is not scheduled at this point. The Ampersand Team will be invited to attend the final demonstration which is to be scheduled in April.

4.7 Sentinel Test

After review and acceptance of the EFA project. EFA will be ran on the sentinel (see test case T13 on page 18 of Test Plan). The sentinel test is performed at regular intervals and emails developers about any failed test. This will serve as automated testing of EFA project in the future.

5 Changes Made After testing

After intense usability testing, the EFA team decided to format the generated SQL using a pretty printer library. The formatted SQL is indented for better readability and thereby increasing the overall usability of the EFA project.