Randa Mohsen

1314785

COMP SCI 3EA3

MUSA AL-HASSY

Individual Project Report

Alloy budgeting model

https://github.com/RandaHacks/BudgetingModel

# **ABSTRACT**

The purpose of this project was to create a Budgeting Model using the software modeling analyzer, Alloy. The Budgeting Model was built with a lot of object oriented structures in mind. Its purpose is to create a model in which transactions, payments, currencies are handles in one place. Alloy was chosen for multiple reasons, which will be discussed in more detail below, including its extremely easy accessibility and automatic analysis techniques.

Table of Contents

[ABSTRACT 2](#_Toc512425716)

[INTRODUCTION 4](#_Toc512425717)

[APPLICATION 5](#_Toc512425718)

[Files 5](#_Toc512425719)

[Code 5](#_Toc512425720)

[TEST CASES 6](#_Toc512425721)

[TOOLS 7](#_Toc512425722)

[Alloy 7](#_Toc512425723)

[**Why?** 7](#_Toc512425724)

[**How?** 7](#_Toc512425725)

[CONCLUSION 9](#_Toc512425726)

[REFERENCES 9](#_Toc512425727)

# **INTRODUCTION**

Inspiration for this project comes from the universal need to budget one’s income and spending. The model will be based around Transactions, Links between transactions, and properties of transactions. A transaction is a transfer of money between a source and a destination. An optional property of a transaction is whether it belongs to a category of luxury, or necessity.

# **APPLICATION**

## Files

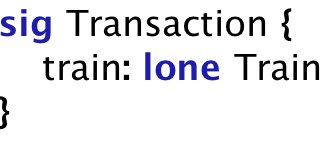
The code for this project can be found in a file called ***transactions.als*** which can be found in the main directory of the GitHub repository found at:

<https://github.com/RandaHacks/BudgetingModel>

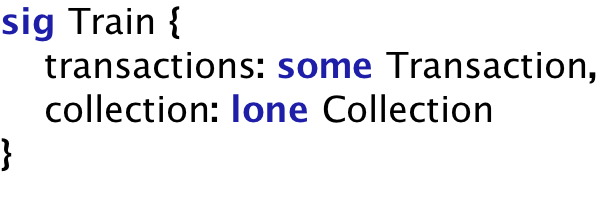
For directions on how to open this file with the Alloy Analyzer, check the “Tools” subsection of this report.

## Code

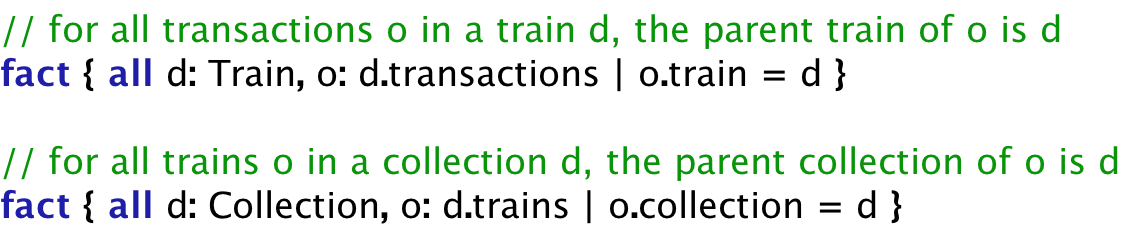
The main code in this file consists of four of the main alloy keywords: signature, fact, predicate, and run. A signature is used to define our main objects of the budgeting model. For example, the following “Transaction” object is a signature that doesn’t extend any other, and has one property.



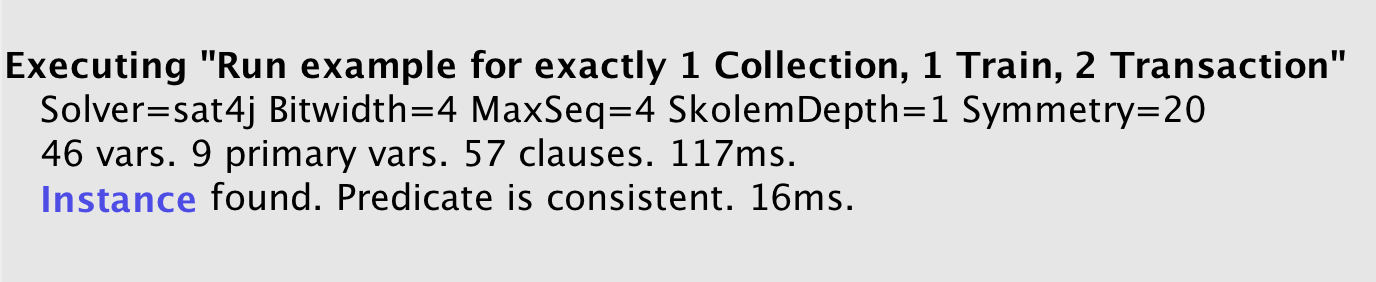
“train” is the name of the property inside the signature, and is of type Train which is defined using another signature. This property is necessary, as it constrains a Transaction to strictly belong to zero or one Train.

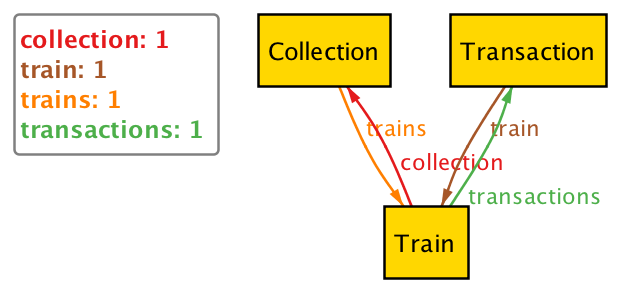


The next alloy keyword uniquely used in this code is: fact. It is used to apply two different rules to the model. These rules are assumptions made by the coder to indicate that they should *always* hold for any instance of the model.



# **TEST CASES**





# **TOOLS**

## **Alloy**

### **Why?**

There are many reasons Alloy was picked to develop this Budgeting Model.

1. **Accessibility** – The Alloy Analyzer is available on the main website, and is downloaded in the form of a .jar file. This makes it an extremely accessible tool as jar files can be run on any operating system for which a JVM exists.
2. **Finite Scope Check** – In order to analyze a model, you must give a specific scope to check within. By using this approach, Alloy can guarantee correctness up to the specified scope.
3. **Infinite Model** – With alloy modeling, you only describe how the components of a system interact, not how many there are. This allows you to expand on your model infinitely as it grows object by object.
4. **Automatic Analysis** – Alloy allows for automatic generation of solutions or counter examples to a specified system. As early as before you save your file, you can execute a run or check command statement in order to receive an instance or counter example, respectively.
5. **Structured Data** – Alloy models are capable of handling complex data structures. For example, trees. Also, while it is not primarily built for this, it can handle object oriented code very similarly to how that of java.

### **How?**

#### Download:

**Step 1-** Go to the main website for Alloy Tools:

http://alloytools.org/

**Step 2-** On the top navigation bar, click the button:

Download

**Step 3-** Choose the third option, as it is the most stable and platform independent version:

allo4.jar

#### Open:

**Step 1-** Save the jar file in an appropriate folder on your local machine.

**Step 2-** Simply double click on the jar file at any time to open it.

**Step 3-** You have two options.

If you’d like to create a model from scratch simply start typing in the blanks screen provided and save your work. Your file will automatically have the ***.als*** extension and can now be opened by any Alloy Analyzer Tool on any machine.

Otherwise, if you’d like to open an existing ***.als*** file then you can either use the “open” button on the navigation bar or go to File -> Open. Navigate to your desired file. Once it is opened in your tool, you have the option to modify it directly or use it as a template to create a new file with a new name.

Note: the tool allows you to open multiple files at once.

#### Run:

The analyzer provides two distinct methods in which to test the correctness of a model. When you create assert statements,

Consider the code from this project as a sample. The main file in which the code for the model lives, is also where run and check statements will be written. A run statement is alloy’s method for providing you with an instance of your model. This instance is not promised to be the least, nor the greatest instance which is possible in your scope.

For example:

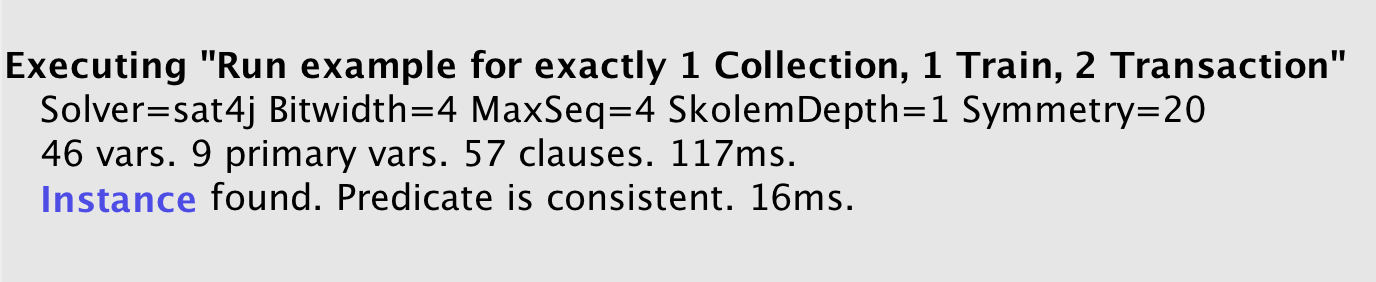
**Step 1:** The following statements are coded in the file



**Step 2:** The button labeled “Exectute” on the navigation bar is selected.



**Step 3:** On the right-hand side of the Analyzer, we see a panel in which execute statements will output information. In this case, the information given is in the following form.



If there is an instance found, you can click on the word “instance”. If not, alloy will indicate “no instance found” and you will have no choice but to modify your model or execute one of your check statements.

# **REFERENCES**

Alloy - http://alloytools.org/  
  
GitHub Repo - https://github.com/RandaHacks/BudgetingModel