POLITECNICO
MILANO 1863

SafeStreets project

Manuel Pedrozo, Tomás Perez Molina

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Manuel Pedrozo, Tomás Perez Molina
Version:	1.0
Date:	November 7, 2019
Download page:	https://github.com/lethanity/PedrozoPerez
Copyright:	Copyright © 2019, Manuel Pedrozo, Tomás Perez Molina – All rights reserved

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The purpose of this document is to provide a description of the SafeStreets system. A detailed explanation of the proposed solution is given, along with the requirements and assumptions made to achieve it.

SafeStreets is a crowd-sourced application that intends to provide users with the possibility to notify authorities when traffic violations occur, and in particular parking violations. With the amount of traffic we are seeing nowadays, it is hard to maintain order throughout the entire city, so the help of the community is more than welcome.

The application allows users to report violations by sending pictures, along with important information, like the date, time and position.

Examples of violations are vehicles parked in the middle of bike lanes or in places reserved for people with disabilities, double parking, and so on.

The system also allows both end users and authorities to access the information gathered, with different levels of visibility depending on the roles.

With the information provided, it is then possible for the municipality to integrate it with their traffic ticket system and automatically issue the corresponding ticket to a reported offender. This will accelerate the whole process, saving time and money to the state and could eventually result in a decrease in violations.

At the same time, the ticketing system can provide information to SafeStreets, which presents the possibility of building statistics such as the most egregious offenders and analyse the effect of the application by looking at the trend in violations.

1.2 Scope

1.2.1 General purpose

As already mentioned, the SafeStreets system is designed to provide users with the ability to report and get information of reported traffic violations through an application.

Any user with a device capable of running the application can sign up to the system, which enables them to access its functionalities.

In order to submit a report, the user needs to fill a form. In it, they have to enter the license plate number of the vehicle committing the violation, the type of violation and at least one photo of the scene, where the license plate of the vehicle can be easily recognized. This data, along with metadata retrieved from the user's device (geographical position, date and time) is then sent to the system.

The system is responsible for analysing the validity of the report. To achieve this, a license plate recognition algorithm is utilized. The output consists of possibly multiple license plates (the picture could include more than one car), along with the certainty of the detection. If multiple license plates are detected, the target of the report is determined by comparing it with the input of the user in the form. After a target license plate is confirmed, the confidence of the detection is evaluated, if it is below a certain threshold, it must pass through a community review. During this process, multiple users willing to participate are shown a cutout of the license plate in the picture and asked to input what they see. If a consensus is reached, then the report is considered valid.

The data collected by the system in relation to reports is to be queried by its users. There are two distinct targets of this functionality: standard users and the municipality. The main difference between the two is that the municipality can access information that should not be freely accessible to everyone because of security and privacy concerns. Through the application, users are capable of visualizing a city map showing where the violations happened. Furthermore, a public API is made available, facilitating data

1.2.2 World and Machine phenomena

To mark the boundaries of the system, here we denote:

- The world phenomena which concern the system (the machine).
- The phenomena internal to the machine from a high level point of view.
- Shared phenomena that cross from the world to the machine or vice versa.

Phenomenon	**Shared**	**Controlled by**
A person commits a traffic violation	N	World
User spots a traffic violation	N	World
User logs in	Y	World
User fills a report form	Y	World
User takes pictures of the traffic violation	Y	World
User submits report	Y	World
Machine analyzes the pictures to find a license plate	N	Machine
Machine accepts the submitted report	Y	Machine
Machine rejects the submitted report	Y	Machine
User wants to find information about traffic violations	N	World
User requests report information in a specific area and time	Y	World
Machine receives the request and filters its stored reports according to the query	N	Machine
Machine answers the request for information	Y	Machine
Authority uses report information to generate traffic tickets	N	World
Machine requests information about traffic tickets to an external traffic ticket system	Y	Machine
Traffic ticket system responds to the information request	Y	World
Machine cross references its stored report information with the traffic ticket system response and analyses it	N	Machine

Table 1: World and Machine phenomena

1.2.3 Goals

G1 - The user is able to report a traffic violation to authorities.

G2 - The user is able to visualize reports in a specified area and time.

G3 - It is possible to query report information in an easily parsable format to allow for data analysis.

G4 - Only authorities are able to access report pictures and license plates.

G5 - The system must protect the chain of custody of the reports.

G6 - Compromised reports are detected and discarded.

G7 - Information about issued tickets provided by the municipality system can be cross referenced with the SafeStreets database and analysed.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- Compromised report: a report that has been modified by an unauthorized agent outside the system boundaries.
- Authority: a local agency whose purpose is, as indicated by the current law, to enforce traffic rules. For example: the police.

1.3.2 Revision history

- Version 1.0: First release.

1.3.3 Reference documents

1.3.4 Document structure

The RASD document is composed of five chapters:

Chapter 1: The problem is introduced. A description of the purpose of the application is given, followed by the scope, where the world and machine phenomena are explained, along with the system goals. Also, definitions are listed to help the reader understand the concepts used.

Chapter 2: An overall description of the product. Including a further detailed description of the product, with the help of class and state diagrams, a description of the main functionalities, the different types of actors that interact with the system, and the domain assumptions considered for solving the problem.

Chapter 3: Specific requirements. This is the main chapter of the document. It includes the external interface requirements, like the user and software interfaces. Scenarios for typical application usage are provided, followed by functional and performance requirements, and the constraints under which the system needs to function. Lastly, the system software attributes are discussed.

Chapter 4: In this chapter a documented Alloy model is presented for a formal analysis of the problem, along with a discussion of the purpose of this model and what it proves.

Chapter 5: Shows the effort spent by each member of the group in the development of the document.

2 Overall Description

2.1 Product perspective

SafeStreets is built from the ground up as a new software application, specifically a mobile application, as required by the functionality that is provided. This mobile app communicates with a server, where the data is analysed and stored. External services are also utilized, particularly a government licence plate registry and a maps API.

As for the SafeStreets system, the domain model is described in the diagram shown in Figure 1. Note that the diagram is not a complete description of the system, but rather a simplified version for easier understanding.

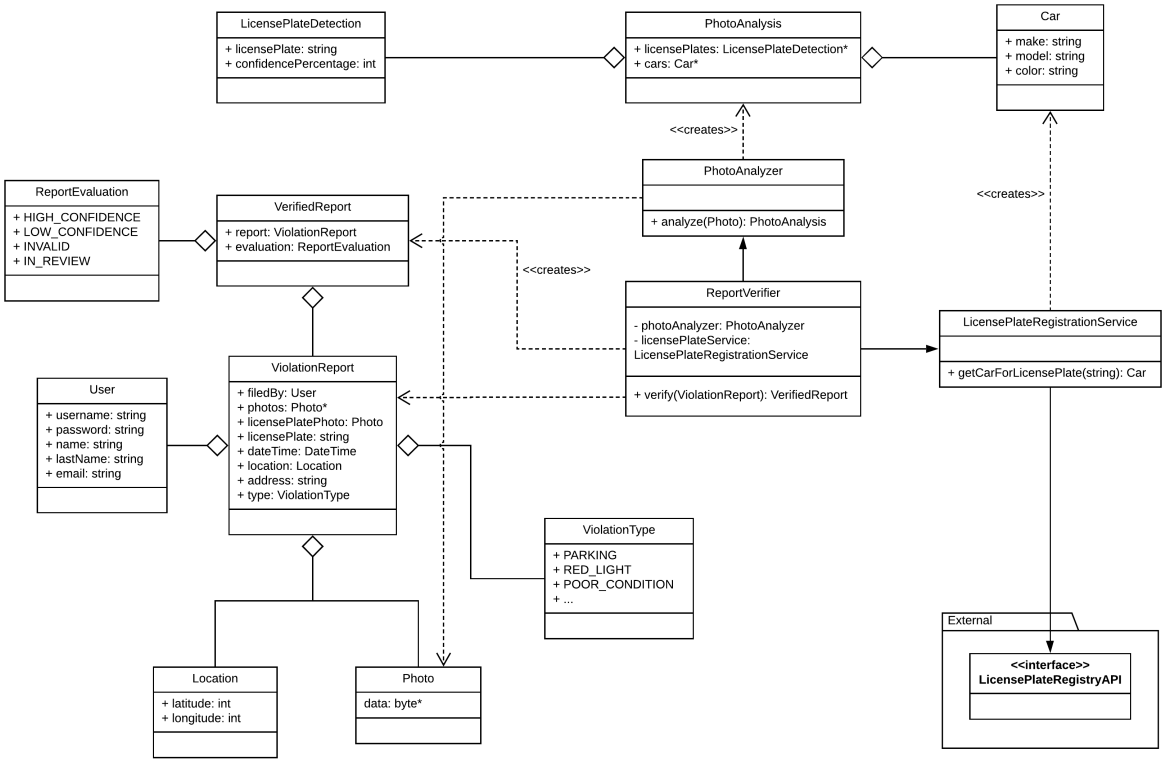

Figure 1: Class diagram.

Inspecting the class diagram, we can see that most of the system revolves around the violation reports and their processing, this is the core of the system. In the state diagram shown in Figure 2, the process of submitting a report is explained.

As observed in the diagram, for submitting a report, the user is required to both fill a formulary with the required information and take photos of the event. After the submission, the SafeStreets system executes an analysis of the data, matching it with the provided image. This can result in either a invalid or a valid report. An invalid report is discarded, with possible measures taken against the account that submitted it. A valid report is saved in the database. When the report is saved, it is made available for the different users to query, either through the mobile application or through the public API.

Further explaining the analysis process Figure 3, the system utilizes a license plate recognition algorithm which output consists of possibly multiple license plates (the picture could include more than one car), along with the certainty of the detection. If multiple license plates are detected, the target of the report is determined by comparing it with the input of the user in the form. After the target license plate is confirmed, the confidence of the detection is evaluated, if it is below a certain threshold, it must

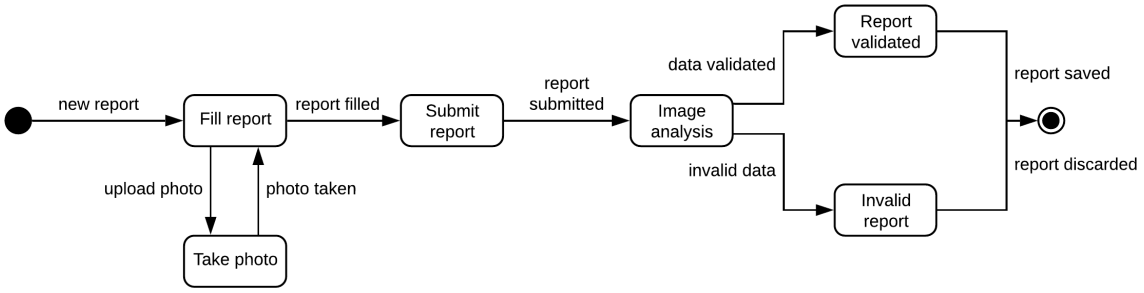

Figure 2: State diagram - Report submission.

pass through a community review. During this process, multiple users willing to participate are shown a cutout of the license plate in the picture and asked to input what they see. If a consensus is reached, then the report is considered valid.

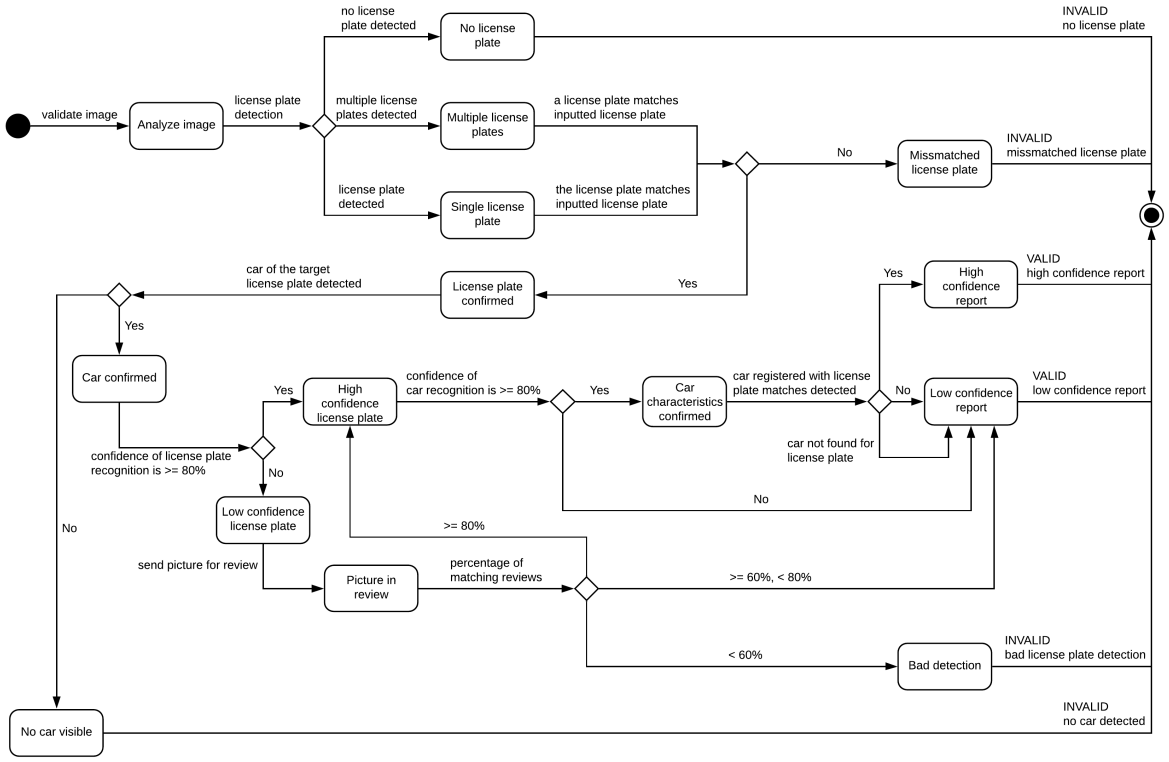

Figure 3: State diagram - Image validation.

2.2 Product functions

The functionality of the system can be divided into three groups. In the following section, these functions are listed and explained, taking into account the already specified goals of the system.

2.2.1 Violation report

The reporting of violations is the main functionality of the system. It allows its registered users to submit a traffic violation report. The user is required to provide data, such as pictures of the violation, the license plate of the vehicle committing the violation and the type of violation. On top of this information, the mobile application will provide the system with metadata which includes the gps location, date and time of the report. After the submission, the system analyses the provided information, checking its integrity. If the report is considered invalid or to have been compromised, it is discarded and the user is flagged as not trustworthy. Otherwise, the report is saved and made available to the rest of the system.

2.2.2 Image analysis

In order to confirm the validity of a report, the system performs an analysis of the submitted images. The pictures are expected to show the vehicle committing the violation, with at least one of them providing a clear view of its license plate. The analysis searches the images for this information and matches the detected license plate with the one provided by the user in the report.

2.2.3 Data querying

Gathered information by SafeStreets can be accessed by all users. There are two ways in which the data can be accessed, via the mobile application or through the public API. In the first case, the mobile application provides users with the ability to see violations in a map, allowing the user to also filter these violations by date and type. On the other hand, the API allows for SafeStreets to be integrated with other third party systems. Users are able to query the available data according to their role and obtain the information in an analysis-friendly format.

2.3 User characteristics

The actors identified as the users of the application are:

- User: Also referred to as the “standard user”. A person that has registered to SafeStreets and is capable of reporting violations, seeing the reports map and reviewing photos.
- Municipality system: A system belonging to the municipality that communicates with SafeStreets through the exposed API. Capable of accessing more information than the standard user.
- Administrator: An employee of SafeStreets that maintains and updates the system.

2.4 Assumptions, dependencies and constraints

D1 - An accurate gps location can be acquired from the device the user is running SafeStreets on.

D2 - The device running SafeStreets has a camera.

D3 - A violation report cannot be canceled

D4 - There is a system available capable of connecting a license plate to characteristics of the car (make, model, color)

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

The following mockups show an approximation of the mobile application.

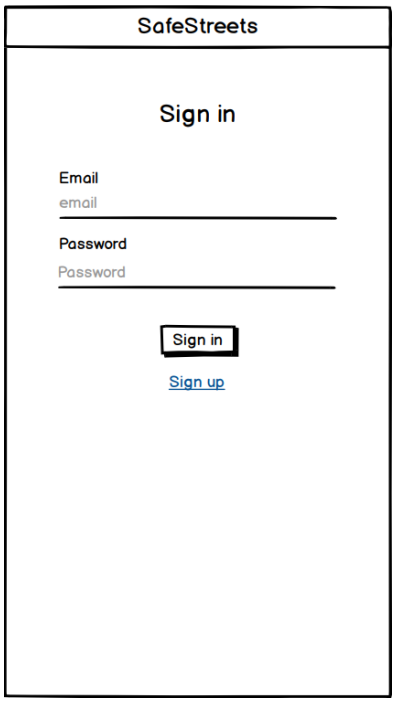

Figure 4: Mockup - Sign in.

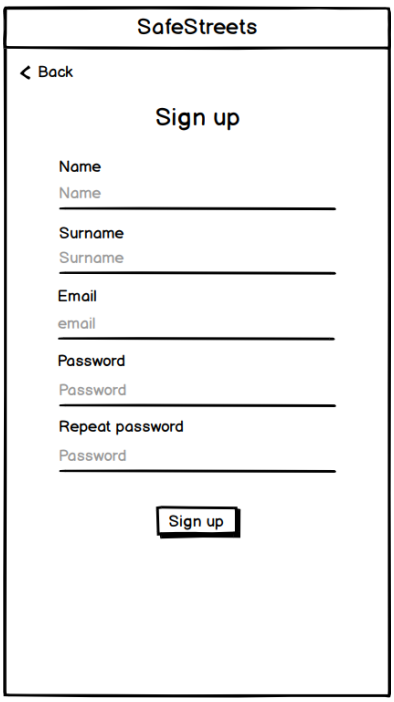

Figure 5: Mockup - Sign up.

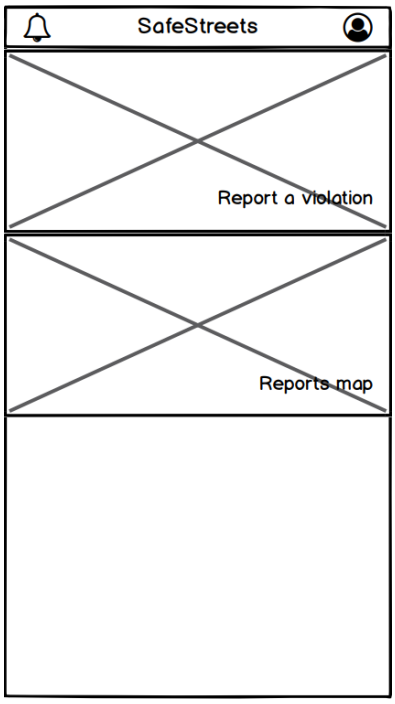

Figure 6: Mockup - Home.

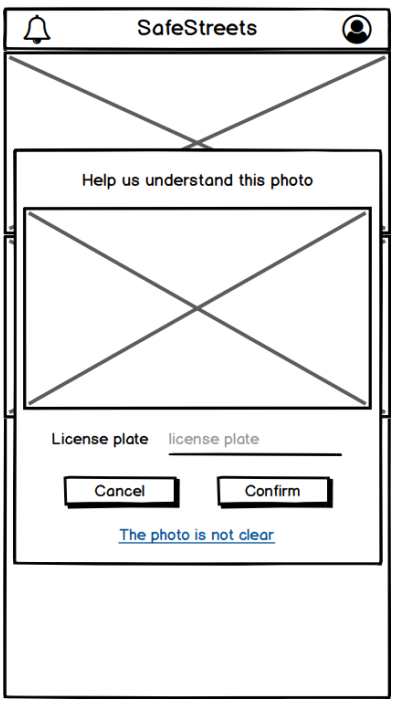

Figure 7: Mockup - Photo review.

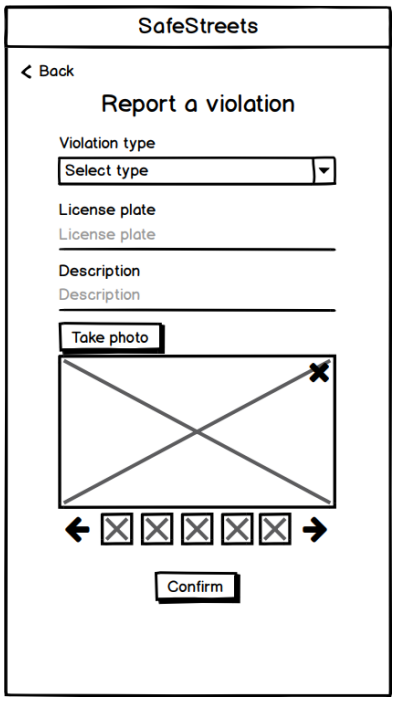

Figure 8: Mockup - Report violation.

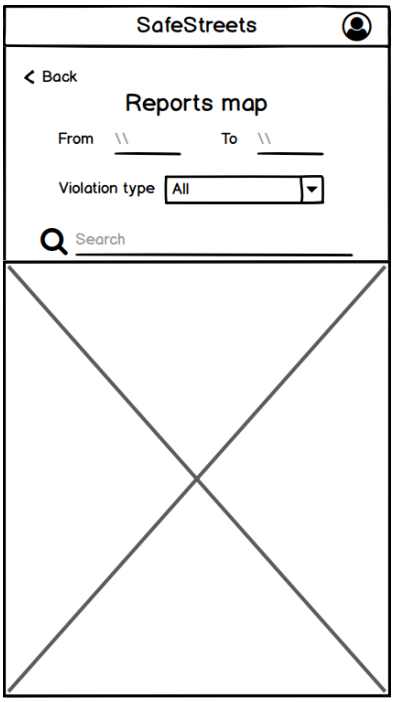

Figure 9: Mockup - Reports map.

3.1.2 Hardware interfaces

3.1.3 Software interfaces

- License plate recognition: it is necessary to be able to recognize the license plate in the picture a report. For this, a good option is OpenALPR, an open source library that can be run server-side (due to performance limitations). A cloud API is also provided, which could be used during early stages of development.

3.1.4 Communication interfaces

3.2 Scenarios

3.2.1 Scenario 1

John has a spot on his street reserved for his garage entrance. During the week, he commutes to work by subway, and leaves the car to his wife, Sarah, who leaves later in the day. It is not unusual for him to find someone blocking the garage when he goes to work in the morning. If the car is still there by the time Sarah needs to leave, she will be late to the office. Before using SafeStreets, he would have to call the police and provide a license plate and address, all while hurrying on his way to work. Now, he can just take a picture and then submit a report while he is sitting on the subway.

3.2.2 Scenario 2

Marco is a data science student at Politecnico di Milano. As he bikes every day to the university, he is familiar with the problem of cars parking in the bike lane. Thanks to the SafeStreets API, he can easily obtain data about it using Python. So, he decides to base his thesis on this topic and analyzes the patterns in violations throughout all of Italy, comparing different cities and their countermeasures.

3.2.3 Scenario 3

Chad's girlfriend broke up with him last week because he was too jealous and would not let her go out with her friends. He is not over this and is really mad at her. All of the sudden he gets a brilliant idea, he takes a picture of the license plate of the girls car and prints it. Then, a couple days later he finds a double parked car, attaches the printed license plate to it and reports the incident with SafeStreets. The system detects that something is wrong with the picture, so it decides not to save the report and marks Chad's account with potential misuse of the application.

3.2.4 Scenario 4

Sally wants to go to the city centre to buy some groceries, but she is not sure if she should take the car or go by bus. She grabs her phone, boots up SafeStreets and checks the vicinity to the supermarket. The app shows a high concentration of badly parked cars, she assumes that there is a lot of traffic and nowhere to park her car properly, so she decides to take the bus.

3.3 Functional requirements

G1 - The user is able to report a traffic violation to authorities

- D1 - An accurate gps location can be acquired from the device the user is running SafeStreets on.
- D2 - The device running SafeStreets has a camera.
- D4 - The metadata of the picture in the violation report is accurate
- R1 - The user must be able to fill out a form providing information about a traffic violation.

G2 - License plates can be recognized from the pictures of the violation report

-

G3 - Roles with different levels of permission are assigned to users and authorities.

-

G4 - The information gathered from the reports is provided to users and authorities according to their role.

-

G5 - The system must protect the chain of custody of the reports

- RX - The system must provide e2e encryption from the device of the user to the server.

G6 - Reports that had their integrity compromised and malicious reports will be detected and discarded.

- D5 - There is a system available capable of connecting a license plate to characteristics of the car (make, model, color).
- RX - SafeStreets must cross reference the information of the car registered under the detected license plate with the information obtained from analyzing the report picture.

G7 - Information about issued tickets provided by the municipality system can be cross referenced with the SafeStreets database and analysed

-

3.3.1 Use cases

Name	Sign up
Actor	User
Entry condition	- The user has installed the application on their device. - The application is running.
Event flow	1. The user presses the “Sign up” button. 2. The user fills the fields with the required data. 3. The user presses the “Confirm” button. 4. The system saves the data.
Exit condition	- The user is successfully registered in the system. - The user is redirected to the login screen.
Exceptions	- The user is already registered in the system. The system warns the user that the email is already in use. - The user did not fill all the required fields. The system marks the empty fields for the user to fill. - The password does not meet the security requirements. The system asks the user to enter another password.

Name	Sign in
Actor	User
Entry condition	- The application is running. - The user is signed up.
Event flow	1. The user presses the “Sign in” button. 2. The user fills the “Username” and “Password” fields. 3. The user presses the “Sign in” button. 4. The system verifies the user credentials.
Exit condition	- The user is successfully signed into the system. - The user is redirected to the home screen.
Exceptions	- The user enters a non matching combination of “username” and “password”. The system shows a warning that “username” and “password” do not match. - The user did not fill all the required fields. The system marks the empty fields for the user to fill.

Name	See profile
Actor	User
Entry condition	- The application is running. - The user is signed in. - The user is in the home screen.
Event flow	1. The user presses the user icon button. 2. The system shows the user information.
Exit condition	- The user information is displayed to the user.

Name	Edit user information
Actor	User
Entry condition	- The application is running. - The user is signed in. - The user is in the home screen.
Event flow	1. The user presses the “edit” button. 2. The user edits the fields they want to change. 3. The user presses the “confirm” button. 4. The system saves the data.
Exit condition	- The user information is successfully updated. - The user profile is in view-only state.
Exceptions	- The user did not fill all the required fields. The system marks the empty fields for the user to fill. - The email is already registered in the system. The system warns the user that the email is already in use.

Name	Submit report
Actor	User
Entry condition	- The application is running. - The user is signed in. - The user is in the home screen. - The user’s GPS is active.
Event flow	1. The user presses the “Report a violation” button. 2. The user fills the fields with the required data. 3. The user presses the “Take photo” button. 4. The user takes a photo of the vehicle committing the violation. 5. The user repeats steps 3 and 4 as desired until the amount of photos reaches the limit. 6. The user presses the “Confirm” button. 7. The system prompts the user to select a photo where the license plate is clearly identifiable. 8. The user selects a photo. 9. The user presses the “Confirm” button. 10. The system submits the report
Exit condition	- The report is successfully submitted.
Exceptions	- The user did not fill all the required fields. The system marks the empty fields for the user to fill. - The user did not take a photo. The system warns the user to take a photo

Name	See reports map
Actor	User
Entry condition	- The application is running. - The user is signed in. - The user is in the home screen.
Event flow	1. The user presses the “Reports map” button. 2. The user fills the “from”, “to” and “type” fields. 3. The user presses the “filter” button. 4. The system shows an interactable map with reports that match the filter.
Exit condition	- The system shows the reports map.
Exceptions	- No reports matching the filter were found. The system shows the empty map.

Name	Review photo
Actor	User
Entry condition	- The application is running. - The user is signed in. - The user is in the home screen.
Event flow	1. The user presses the review photo button. 2. The user fills the “license plate” field. 3. The user presses the “confirm” button. 4. The system saves the review.
Exit condition	- The review is saved in the system. - The system shows another photo to review.
Exceptions	- The user did not fill the “license plate” field. The system marks the empty field for the user to fill. - The user presses the “The photo is not clear” button. The system shows another photo to review.

3.4 Performance requirements

3.5 Design constraints

3.5.1 Standards compliance

3.5.2 Hardware limitations

For use of the mobile application, a smartphone or tablet with the following specifications is required:

- Internet connection (Wi-Fi/4G/3G/2G)
- GPS
- Camera
- Android or iOS operating system

3.6 Software system attributes

3.6.1 Reliability

The application is expected to run continuously with no downtime. But given that the system is by no means critical to any of its users, exceptions to this requirement are tolerated. In terms of stored data, the system is required to be fault tolerant, which means that the data needs to be replicated and stored in more than one location.

3.6.2 Availability

Although minimal downtime is tolerated, the system is expected to be available 99.9% of the time. Because of this, some redundancy is to be provided in the application servers.

3.6.3 Security

The system manages sensitive user data, which requires confidentiality. Information like passwords is encrypted before being stored in the database. Measures are to be taken for the protection of servers and databases from both external attacks and hardware malfunctions.

3.6.4 Maintainability

The application is meant to be continuously worked and improved upon, possibly by different teams. This means that good design and documentation are required to facilitate its maintainability.

3.6.5 Interoperability

The system both utilizes services provided by other systems and acts as a service provider. It needs to be compliant with standards for information exchange between systems and provide a clear interface for external users.

4 Formal Analysis Using Alloy

The focus of the analysis is the processing of the reports submitted. A diagram of this can be found in [Figure 3 - State Diagram]. By modeling it using Alloy, we can verify the following points:

- All possible scenarios of a report analysis are covered by our established definitions.
- There is no overlap of report definitions. An overlap would make the definitions ambiguous and the final result would ultimately depend on the implementation.
- All definitions of the result of a report analysis are possible with the given assumptions and constraints.

Clarifications
Some aspects of the report are ignored (such as time, location and the user that submitted it) as they are not constrained in any way and do not affect the analysis of the report.
The rest of the system is not contemplated as it is fairly straightforward and its properties can be trivially checked (for example, no two users share the same username/email).

4.1 The model

```
module model

---------------------------------- DETECTION ----------------------------------

sig LicensePlate, Car {}

sig Confidence {
        rate: Int
} {
        // Confidence is between 0 and 100%. But not exactly 0.
        rate > 0 and rate ≤ 10
}

sig DetectedLicensePlate {
        licensePlate: LicensePlate,
        confidence: Confidence
}

sig DetectedCar {
        car: Car,
        confidence: Confidence
}

// Detection of a license plate and the car on which it is set on.
sig Detection {
        car: lone DetectedCar,
        licensePlate: lone DetectedLicensePlate
}

fact DetectionsCannotLackCarAndLicensePlate {
        no d: Detection | no d.car and no d.licensePlate
}

fun Detection.getLicensePlateWithCar: set LicensePlate -> Car {
        this.licensePlate.licensePlate -> this.car.car
}

pred licensePlateDetectionIsTrustworthy [d: Detection] {
        d.licensePlate.confidence.rate ≥ 8
}

pred carDetectionIsTrustworthy [d: Detection] {
        d.car.confidence.rate ≥ 8
}
```



```
------------------------------- PHOTO ANALYSIS -------------------------------

sig Photo {}

/*
        An analyzed photo, where multiple cars and their license plates
        could be detected.
*/
sig AnalyzedPhoto {
        photo: Photo,
        detected: set Detection
}

fact NoCarIsDetectedTwice {
        no disj d1, d2: AnalyzedPhoto.detected |
                some d1.car and
                some d2.car and
                d1.car.car = d2.car.car
}

fact NoLicensePlateIsDetectedTwice {
        no disj d1, d2: AnalyzedPhoto.detected |
                some d1.licensePlate and
                some d2.licensePlate and
                d1.licensePlate.licensePlate = d2.licensePlate.licensePlate
}

fun AnalyzedPhoto.getAnalyzedPhotoLicensePlates : set LicensePlate {
        this.detected.(Detection <: licensePlate).(DetectedLicensePlate <: licensePlate)
}

/*
        Repeating the image analysis over a photo gives the same result.
        There cannot be two analysis of the same photo with a different
        set of detected license plates.
*/
fact ImageAnalysisAlwaysReturnsTheSameResult {
        all p, p': AnalyzedPhoto | p.photo = p'.photo implies p.detected = p'.detected
}

---------------------------------- REPORTS -----------------------------------

sig ReportSubmission {
        licensePlate: LicensePlate,
        photos: set Photo,
        licensePlatePhoto: Photo
} {
        licensePlatePhoto in photos
}

sig AnalyzedReport {
        submission: ReportSubmission,
        analyzedPhoto: AnalyzedPhoto
} {
        analyzedPhoto.photo = submission.licensePlatePhoto
}

fun AnalyzedReport.getTargetDetection : set Detection {
        let
                targetLP = this.submission.licensePlate,
                lpDetections = this.analyzedPhoto.detected <: licensePlate {
                        lpDetections.(DetectedLicensePlate <: licensePlate).targetLP
        }
}

fun AnalyzedReport.getTargetCar : set Car {
        getTargetDetection[this].car.car
}

--------------------------- LICENSE PLATE REGISTRY ---------------------------
```



```
// Placeholder for the license plate registration service API.
one sig LicensePlateRegistry {
        registration: LicensePlate -> Car
}

/*
        Every car is registered under at most 1 license plate.
        Every license plate is registered for at most 1 car.

        There could be cases where a car or license plate is not registered:
                - Bad detection
                - Fake license plate
*/
fact NoRepeatedRegistrations {
        all c: Car | lone LicensePlateRegistry.registration.c
        all l: LicensePlate | lone getCarRegisteredUnderLicensePlate[l]
}

fun getCarRegisteredUnderLicensePlate [l: LicensePlate] : set Car {
         l.(LicensePlateRegistry.registration)
}

pred detectionLicensePlateCarMatchIsTrustworthy [d: Detection] {
        let detectedLicensePlateToCar = getLicensePlateWithCar[d] {
                some detectedLicensePlateToCar
                detectedLicensePlateToCar in LicensePlateRegistry.registration
        }
}

pred noCarRegisteredForLicensePlate [l: LicensePlate] {
        no getCarRegisteredUnderLicensePlate[l]
}

---------------------------------- REVIEWS ------------------------------------

sig Review {
        detection: Detection,
        matchPercentage: Int
} {
        // Percentage between 0 and 100%.
        matchPercentage ≥ 0 and matchPercentage ≤ 10
}

fact OnlyReviewLowConfidenceLicensePlateDetections {
        no r: Review | licensePlateDetectionIsTrustworthy[r.detection]
}

fact OnlyReviewDetectionsWithLicensePlateAndCar {
        all d: Review.detection | some d.car and some d.licensePlate
}

fact OnlyReviewDetectionsOfTarget {
        Review.detection in getTargetDetection[AnalyzedReport]
}

fact NoRepeatedReviews {
        no disj r1, r2: Review | r1.detection = r2.detection
}

pred badDetectionReview [r: Review] {
        some r
        r.matchPercentage < 6
}

pred acceptableDetectionReview [r: Review] {
        r.matchPercentage ≥ 6 and r.matchPercentage < 8
}

pred highConfidenceDetectionReview [r: Review] {
        r.matchPercentage ≥ 8
}
```



```
---------------------------------- EXTRA -------------------------------------

fact NoDanglingData {
        // All reports are analyzed.
        ReportSubmission = AnalyzedReport.submission

        // All photos come from submissions.
        Photo = ReportSubmission.photos

        // LicensePlates are detected, submitted, or in the license plate registry.
        LicensePlate =
                DetectedLicensePlate.licensePlate +
                ReportSubmission.licensePlate +
                LicensePlateRegistry.registration.Car

        // Cars are detected or in the license plate registry.
        Car = DetectedCar.car + LicensePlate.(LicensePlateRegistry.registration)

        // Detections come from a photo analysis.
        Detection = AnalyzedPhoto.detected

        // Confidence is attached to a detection.
        Confidence = DetectedLicensePlate.confidence + DetectedCar.confidence

        // Only photos from a report are analyzed.
        AnalyzedPhoto = AnalyzedReport.analyzedPhoto

        // Detected license plates come from a photo detection.
        DetectedLicensePlate = Detection.licensePlate

        // Detected cars come from a photo detection.
        DetectedCar = Detection.car
}

open model
module reportDefinitions

---------------------------- MAIN DEFINITIONS ----------------------------

pred reportIsHighConfidence [r: AnalyzedReport] {
        reportHasConfirmedLicensePlate[r]
        reportHasConfirmedCar[r]
        reportHasHighConfidenceLicensePlate[r]
        reportHasConfirmedCarCharacteristics[r]
        detectionLicensePlateCarMatchIsTrustworthy[getTargetDetection[r]]
}

pred reportIsLowConfidence [r: AnalyzedReport] {
        reportHasConfirmedLicensePlate[r]
        reportHasConfirmedCar[r]
        !reportHasBadlyDetectedLicensePlate[r]
        (
                reportHasAcceptableLicensePlateReview[r] or
                !carDetectionIsTrustworthy[getTargetDetection[r]] or
                noCarRegisteredForLicensePlate[r.submission.licensePlate] or
                !detectionLicensePlateCarMatchIsTrustworthy[getTargetDetection[r]]
        )
}

pred reportHasNoDetectedLicensePlate [r: AnalyzedReport] {
        no getAnalyzedPhotoLicensePlates[r.analyzedPhoto]
}

pred reportHasNonMatchingLicensePlates [r: AnalyzedReport] {
        let licensePlates = getAnalyzedPhotoLicensePlates[r.analyzedPhoto] {
                some licensePlates
                ! r.submission.licensePlate in licensePlates
        }
}

pred reportHasBadlyDetectedLicensePlate [r: AnalyzedReport] {
        badDetectionReview[(Review <: detection).(getTargetDetection[r])]
```



```
}

pred reportHasNoDetectedCarForLicensePlate [r: AnalyzedReport] {
        reportHasConfirmedLicensePlate[r]
        no getTargetCar[r]
}

pred reportIsInReview [r: AnalyzedReport] {
        reportHasConfirmedLicensePlate[r]
        reportHasConfirmedCar[r]
        !licensePlateDetectionIsTrustworthy[getTargetDetection[r]]
        !reportHasReview[r]
}

--------------------------- UTILITY PREDICATES ---------------------------

pred reportHasConfirmedLicensePlate [r: AnalyzedReport]  {
        r.submission.licensePlate in getAnalyzedPhotoLicensePlates[r.analyzedPhoto]
}

pred reportHasAcceptableLicensePlateReview [r: AnalyzedReport] {
        acceptableDetectionReview[(Review <: detection).(getTargetDetection[r])]
}

pred reportHasHighConfidenceLicensePlateReview [r: AnalyzedReport] {
        highConfidenceDetectionReview[(Review <: detection).(getTargetDetection[r])]
}

pred reportHasConfirmedCar [r: AnalyzedReport] {
        some getTargetCar[r]
}

pred reportHasHighConfidenceLicensePlate [r: AnalyzedReport] {
        licensePlateDetectionIsTrustworthy[getTargetDetection[r]] or
        reportHasHighConfidenceLicensePlateReview[r]
}

pred reportHasConfirmedCarCharacteristics [r: AnalyzedReport] {
        carDetectionIsTrustworthy[getTargetDetection[r]]
}

pred reportHasReview [r: AnalyzedReport] {
        some Review.detection & getTargetDetection[r]
}
```

4.2 Assertions and checks

```
module assertions
open model
open reportDefinitions

/*
        Report definitions do not overlap, meaning that a given Submission
        can be classified under only one of the established definitions.
*/
assert ReportDefinitionsAreDisjointed {
        all r: AnalyzedReport {
                reportIsHighConfidence[r] implies
                        !reportIsLowConfidence[r] and
                        !reportHasNoDetectedLicensePlate[r] and
                        !reportHasNonMatchingLicensePlates[r] and
                        !reportHasBadlyDetectedLicensePlate[r] and
                        !reportHasNoDetectedCarForLicensePlate[r]
                        !reportIsInReview[r]
                and
                reportIsHighConfidence[r] implies
                        !reportIsLowConfidence[r] and
                        !reportHasNoDetectedLicensePlate[r] and
                        !reportHasNonMatchingLicensePlates[r] and
                        !reportHasBadlyDetectedLicensePlate[r] and
                        !reportHasNoDetectedCarForLicensePlate[r]
                        !reportIsInReview[r]
```



```
                and
                reportHasNoDetectedLicensePlate[r] implies
                        !reportIsHighConfidence[r] and
                        !reportIsLowConfidence[r] and
                        !reportHasNonMatchingLicensePlates[r] and
                        !reportHasBadlyDetectedLicensePlate[r] and
                        !reportHasNoDetectedCarForLicensePlate[r]
                and
                reportHasNonMatchingLicensePlates[r] implies
                        !reportIsHighConfidence[r] and
                        !reportIsLowConfidence[r] and
                        !reportHasNoDetectedLicensePlate[r] and
                        !reportHasBadlyDetectedLicensePlate[r] and
                        !reportHasNoDetectedCarForLicensePlate[r]
                        !reportIsInReview[r]
                and
                reportHasBadlyDetectedLicensePlate[r] implies
                        !reportIsHighConfidence[r] and
                        !reportIsLowConfidence[r] and
                        !reportHasNoDetectedLicensePlate[r] and
                        !reportHasNonMatchingLicensePlates[r] and
                        !reportHasNoDetectedCarForLicensePlate[r]
                        !reportIsInReview[r]
                and
                reportHasNoDetectedCarForLicensePlate[r] implies
                        !reportIsHighConfidence[r] and
                        !reportIsLowConfidence[r] and
                        !reportHasNoDetectedLicensePlate[r] and
                        !reportHasNonMatchingLicensePlates[r] and
                        !reportHasBadlyDetectedLicensePlate[r]
                        !reportIsInReview[r]
                and
                reportIsInReview[r] implies
                        !reportIsHighConfidence[r] and
                        !reportIsLowConfidence[r] and
                        !reportHasNoDetectedLicensePlate[r] and
                        !reportHasNonMatchingLicensePlates[r] and
                        !reportHasBadlyDetectedLicensePlate[r]   and
                        !reportHasNoDetectedCarForLicensePlate[r]
        }
}

check ReportDefinitionsAreDisjointed for 6 but 5 Int

/*
        Ensure that all possible variations of a report are covered by our
        established definitions.
*/
assert AllReportCasesAreCovered {
        no r: AnalyzedReport |
                !reportIsHighConfidence[r] and
                !reportIsLowConfidence[r] and
                !reportHasNoDetectedLicensePlate[r] and
                !reportHasNonMatchingLicensePlates[r] and
                !reportHasBadlyDetectedLicensePlate[r] and
                !reportHasNoDetectedCarForLicensePlate[r] and
                !reportIsInReview[r]
}

check AllReportCasesAreCovered for 10 but 5 Int
```



```
Executing "Check ReportDefinitionsAreDisjointed for 6 but 5 int"
  Solver=sat4j Bitwidth=5 MaxSeq=6 SkolemDepth=1 Symmetry=20
  23062 vars. 1002 primary vars. 61151 clauses. 227ms.
  No counterexample found. Assertion may be valid. 11229ms.

Executing "Check AllReportCasesAreCovered for 10 but 5 int"
  Solver=sat4j Bitwidth=5 MaxSeq=10 SkolemDepth=1 Symmetry=20
  53086 vars. 2270 primary vars. 142849 clauses. 186ms.
  No counterexample found. Assertion may be valid. 2286ms.

2 commands were executed. The results are:
  #1: No counterexample found. ReportDefinitionsAreDisjointed may be valid.
  #2: No counterexample found. AllReportCasesAreCovered may be valid.
```

Figure 10: Alloy - Assertion results.

4.3 World generation

```
module worldgen
open model
open reportDefinitions

pred ReportWithNoDetectedLicensePlateExists {
        some r: AnalyzedReport | reportHasNoDetectedLicensePlate[r]
}

run ReportWithNoDetectedLicensePlateExists for 2 but 5 Int, 1 AnalyzedReport

pred ReportWithNonMatchingLicensePlatesExists {
        some r: AnalyzedReport | reportHasNonMatchingLicensePlates[r]
}

run ReportWithNonMatchingLicensePlatesExists for 2 but 5 Int, 1 AnalyzedReport

pred ReportWithBadlyDetectedLicensePlateExists {
        some r: AnalyzedReport | reportHasBadlyDetectedLicensePlate[r]
}

run ReportWithBadlyDetectedLicensePlateExists for 2 but 5 Int, 1 AnalyzedReport

pred ReportWithNoDetectedCarForLicensePlateExists {
        some r: AnalyzedReport | reportHasNoDetectedCarForLicensePlate[r]
}

run ReportWithNoDetectedCarForLicensePlateExists for 2
        but 5 Int, 1 AnalyzedReport

pred ReportInReviewExists{
        some r: AnalyzedReport | reportIsInReview[r]
}

run ReportInReviewExists for 2 but 5 Int, 1 AnalyzedReport

pred HighConfidenceReportWithoutReviewExists {
        some r: AnalyzedReport | reportIsHighConfidence[r] and !reportHasReview[r]
}

run HighConfidenceReportWithoutReviewExists for 2 but 5 Int, 1 AnalyzedReport
```



```
pred HighConfidenceReportWithReviewExists {
        some r: AnalyzedReport | reportIsHighConfidence[r] and reportHasReview[r]
}

run HighConfidenceReportWithReviewExists for 2 but 5 Int, 1 AnalyzedReport

pred LowConfidenceReportWithoutReviewExists {
        some r: AnalyzedReport | reportIsLowConfidence[r] and !reportHasReview[r]
}

run LowConfidenceReportWithoutReviewExists for 2 but 5 Int, 1 AnalyzedReport

pred LowConfidenceReportWithReviewExists {
        some r: AnalyzedReport | reportIsLowConfidence[r] and reportHasReview[r]
}

run LowConfidenceReportWithReviewExists for 2 but 5 Int, 1 AnalyzedReport

pred reportHasMultipleLicensePlates [r: AnalyzedReport] {
        #getAnalyzedPhotoLicensePlates[r.analyzedPhoto] > 1
}

pred HighConfidenceReportWithMultipleLicensePlatesExists {
        some r: AnalyzedReport |
                reportIsHighConfidence[r] and reportHasMultipleLicensePlates[r]
}

run HighConfidenceReportWithMultipleLicensePlatesExists for 2
        but 5 Int, 4 LicensePlate, 4 DetectedLicensePlate, 1 AnalyzedReport

pred LowConfidenceReportWithMultipleLicensePlatesExists {
        some r: AnalyzedReport |
                reportIsLowConfidence[r] and reportHasMultipleLicensePlates[r]
}

run LowConfidenceReportWithMultipleLicensePlatesExists for 2
        but 5 Int, 4 LicensePlate, 4 DetectedLicensePlate, 1 AnalyzedReport

pred LowConfidenceReportBecauseOfAcceptableReviewExists {
        some r: AnalyzedReport |
                reportIsLowConfidence[r] and
                reportHasAcceptableLicensePlateReview[r] and
                carDetectionIsTrustworthy[getTargetDetection[r]] and
                !noCarRegisteredForLicensePlate[r.submission.licensePlate] and
                detectionLicensePlateCarMatchIsTrustworthy[getTargetDetection[r]]
}

run LowConfidenceReportBecauseOfAcceptableReviewExists for 2
        but 5 Int, 1 AnalyzedReport

pred LowConfidenceReportBecauseOfBadCarDetectionExists {
        some r: AnalyzedReport |
                reportIsLowConfidence[r] and
                !carDetectionIsTrustworthy[getTargetDetection[r]] and
                !noCarRegisteredForLicensePlate[r.submission.licensePlate] and
                detectionLicensePlateCarMatchIsTrustworthy[getTargetDetection[r]] and
                !reportHasAcceptableLicensePlateReview[r]
}

run LowConfidenceReportBecauseOfBadCarDetectionExists for 2
        but 5 Int, 1 AnalyzedReport

pred LowConfidenceReportBecauseOfNoCarRegisteredForLicensePlateExists {
        some r: AnalyzedReport |
                reportIsLowConfidence[r] and
                noCarRegisteredForLicensePlate[r.submission.licensePlate] and
                carDetectionIsTrustworthy[getTargetDetection[r]] and
                !reportHasAcceptableLicensePlateReview[r]
}

run LowConfidenceReportBecauseOfNoCarRegisteredForLicensePlateExists for 2
        but 5 Int, 1 AnalyzedReport
```



```
pred LowConfidenceReportBecauseOfBadCarAndLicensePlateMatchExists {
        some r: AnalyzedReport |
                reportIsLowConfidence[r] and
                !detectionLicensePlateCarMatchIsTrustworthy[getTargetDetection[r]] and
                !noCarRegisteredForLicensePlate[r.submission.licensePlate] and
                carDetectionIsTrustworthy[getTargetDetection[r]] and
                !reportHasAcceptableLicensePlateReview[r]
}

run LowConfidenceReportBecauseOfBadCarAndLicensePlateMatchExists for 2
        but 5 Int, 1 AnalyzedReport
```



```
Executing "Run ReportWithNoDetectedLicensePlateExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  3884 vars. 208 primary vars. 10532 clauses. 9ms.
  Instance found. Predicate is consistent. 22ms.

Executing "Run ReportWithNonMatchingLicensePlatesExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  3896 vars. 208 primary vars. 10586 clauses. 9ms.
  Instance found. Predicate is consistent. 16ms.

Executing "Run ReportWithBadlyDetectedLicensePlateExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  4359 vars. 208 primary vars. 12131 clauses. 14ms.
  Instance found. Predicate is consistent. 24ms.

Executing "Run ReportWithNoDetectedCarForLicensePlateExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  3926 vars. 208 primary vars. 10610 clauses. 8ms.
  Instance found. Predicate is consistent. 17ms.

Executing "Run ReportInReviewExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  4390 vars. 208 primary vars. 12174 clauses. 8ms.
  Instance found. Predicate is consistent. 16ms.

Executing "Run HighConfidenceReportWithoutReviewExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  5316 vars. 208 primary vars. 15218 clauses. 15ms.
  Instance found. Predicate is consistent. 23ms.

Executing "Run HighConfidenceReportWithReviewExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  5316 vars. 208 primary vars. 15229 clauses. 10ms.
  Instance found. Predicate is consistent. 31ms.

Executing "Run LowConfidenceReportWithoutReviewExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  4890 vars. 208 primary vars. 13799 clauses. 10ms.
  Instance found. Predicate is consistent. 17ms.

Executing "Run LowConfidenceReportWithReviewExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  4890 vars. 208 primary vars. 13810 clauses. 15ms.
  Instance found. Predicate is consistent. 25ms.

Executing "Run HighConfidenceReportWithMultipleLicensePlatesExists for 2 but 5 int, 4 LicensePlate, 4 DetectedLicensePlate, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  5885 vars. 240 primary vars. 16504 clauses. 14ms.
  Instance found. Predicate is consistent. 29ms.

Executing "Run LowConfidenceReportWithMultipleLicensePlatesExists for 2 but 5 int, 4 LicensePlate, 4 DetectedLicensePlate, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  5459 vars. 240 primary vars. 15081 clauses. 11ms.
  Instance found. Predicate is consistent. 30ms.

Executing "Run LowConfidenceReportBecauseOfAcceptableReviewExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  4887 vars. 208 primary vars. 13795 clauses. 11ms.
  Instance found. Predicate is consistent. 22ms.

Executing "Run LowConfidenceReportBecauseOfBadCarDetectionExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  4887 vars. 208 primary vars. 13795 clauses. 12ms.
  Instance found. Predicate is consistent. 24ms.

Executing "Run LowConfidenceReportBecauseOfNoCarRegisteredForLicensePlateExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  4887 vars. 208 primary vars. 13795 clauses. 15ms.
  Instance found. Predicate is consistent. 27ms.

Executing "Run LowConfidenceReportBecauseOfBadCarAndLicensePlateMatchExists for 2 but 5 int, 1 AnalyzedReport"
  Solver=sat4j Bitwidth=5 MaxSeq=2 SkolemDepth=1 Symmetry=20
  4887 vars. 208 primary vars. 13795 clauses. 10ms.
  Instance found. Predicate is consistent. 26ms.

15 commands were executed. The results are:
  #1: Instance found. ReportWithNoDetectedLicensePlateExists is consistent.
  #2: Instance found. ReportWithNonMatchingLicensePlatesExists is consistent.
  #3: Instance found. ReportWithBadlyDetectedLicensePlateExists is consistent.
  #4: Instance found. ReportWithNoDetectedCarForLicensePlateExists is consistent.
  #5: Instance found. ReportInReviewExists is consistent.
  #6: Instance found. HighConfidenceReportWithoutReviewExists is consistent.
  #7: Instance found. HighConfidenceReportWithReviewExists is consistent.
  #8: Instance found. LowConfidenceReportWithoutReviewExists is consistent.
  #9: Instance found. LowConfidenceReportWithReviewExists is consistent.
  #10: Instance found. HighConfidenceReportWithMultipleLicensePlatesExists is consistent.
  #11: Instance found. LowConfidenceReportWithMultipleLicensePlatesExists is consistent.
  #12: Instance found. LowConfidenceReportBecauseOfAcceptableReviewExists is consistent.
  #13: Instance found. LowConfidenceReportBecauseOfBadCarDetectionExists is consistent.
  #14: Instance found. LowConfidenceReportBecauseOfNoCarRegisteredForLicensePlateExists is consistent.
  #15: Instance found. LowConfidenceReportBecauseOfBadCarAndLicensePlateMatchExists is consistent.
```

Figure 11: Alloy - World generation results.

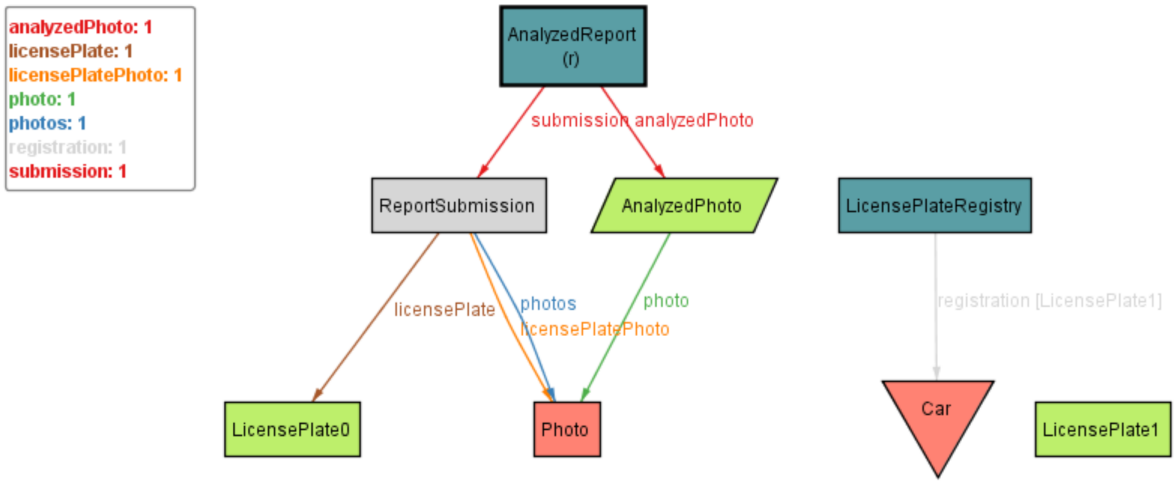

Figure 12: Alloy - World: Report with no detected license plate.

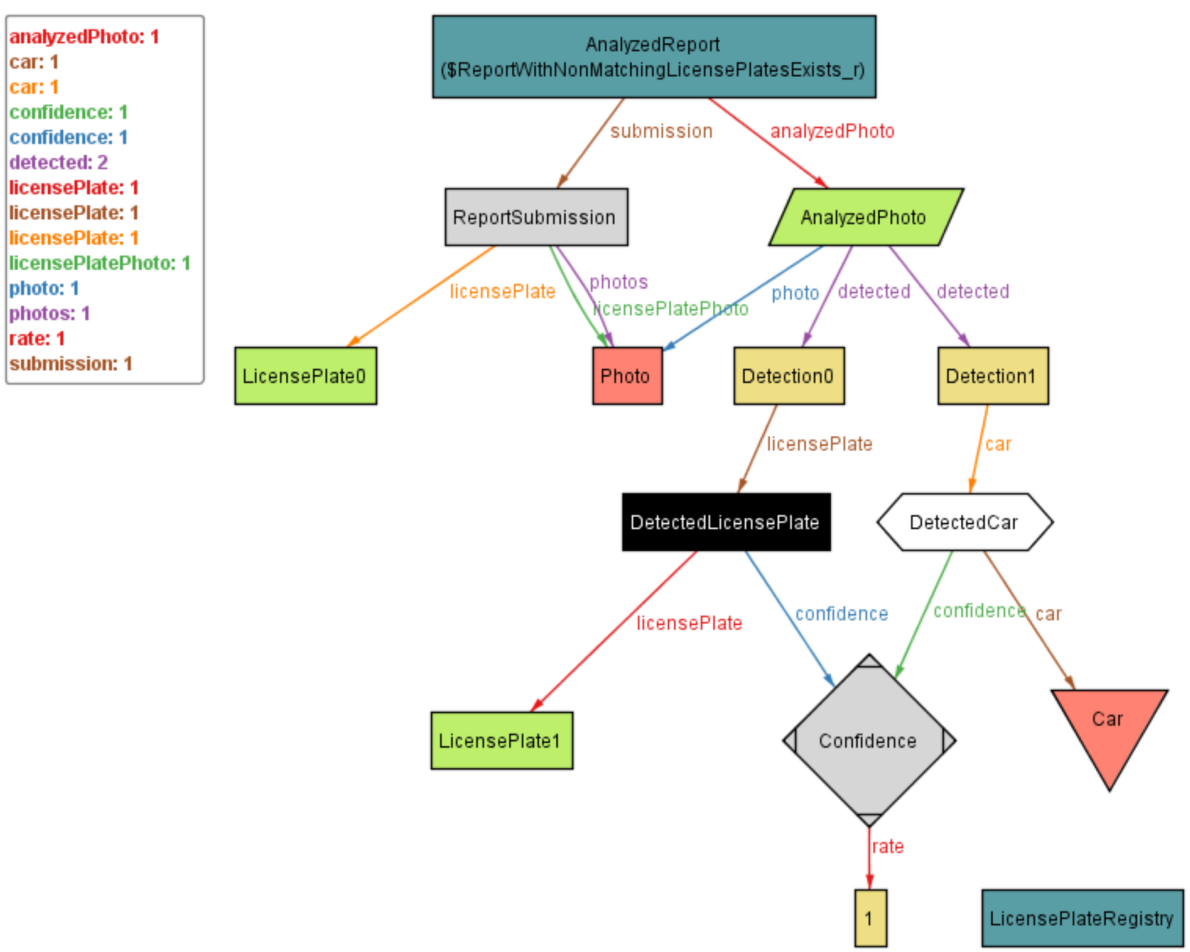

Figure 13: Alloy - World: Report with non matching detected and submitted license plates.

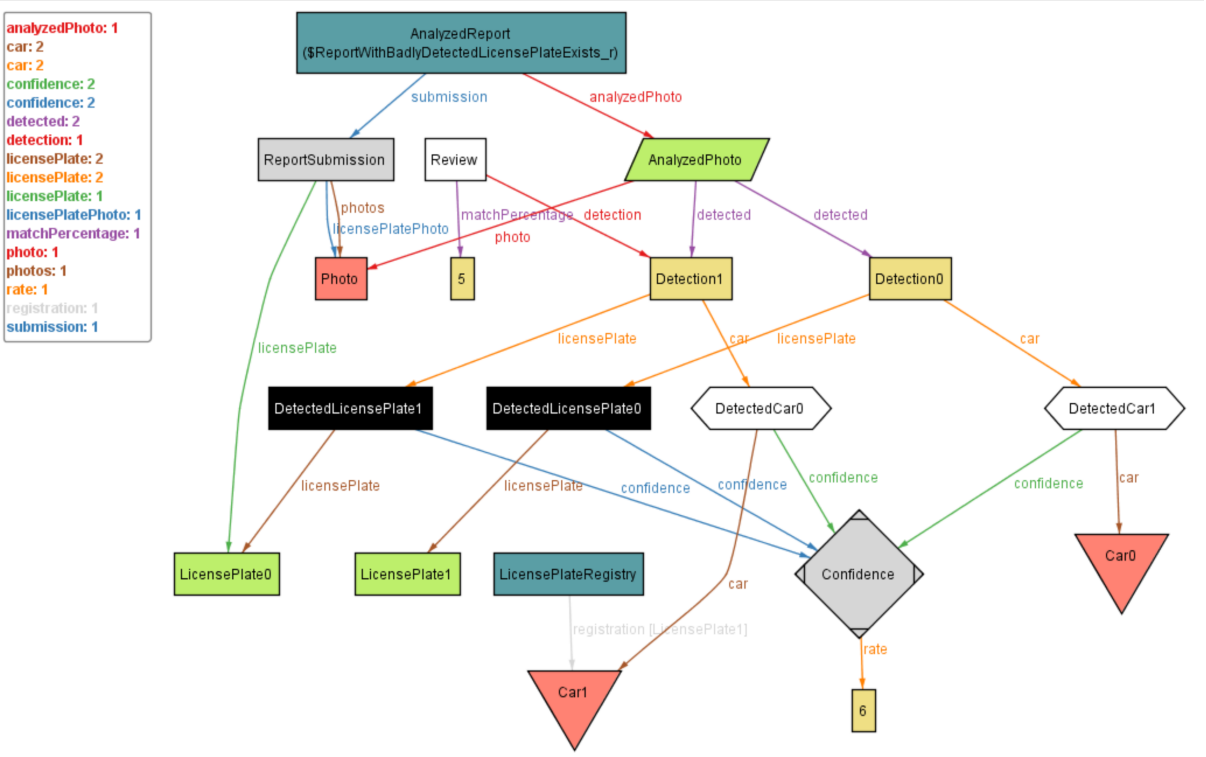

Figure 14: Alloy - World: Report with badly detected license plate.

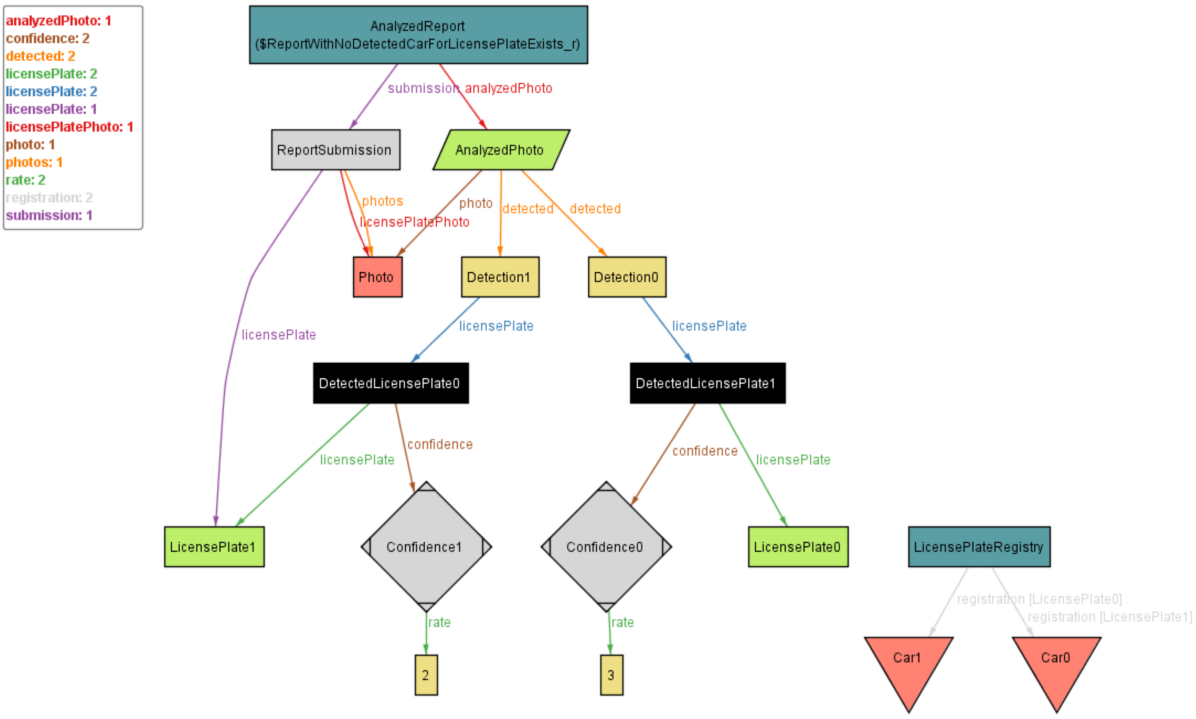

Figure 15: Alloy - World: Report with no detected car for license plate.

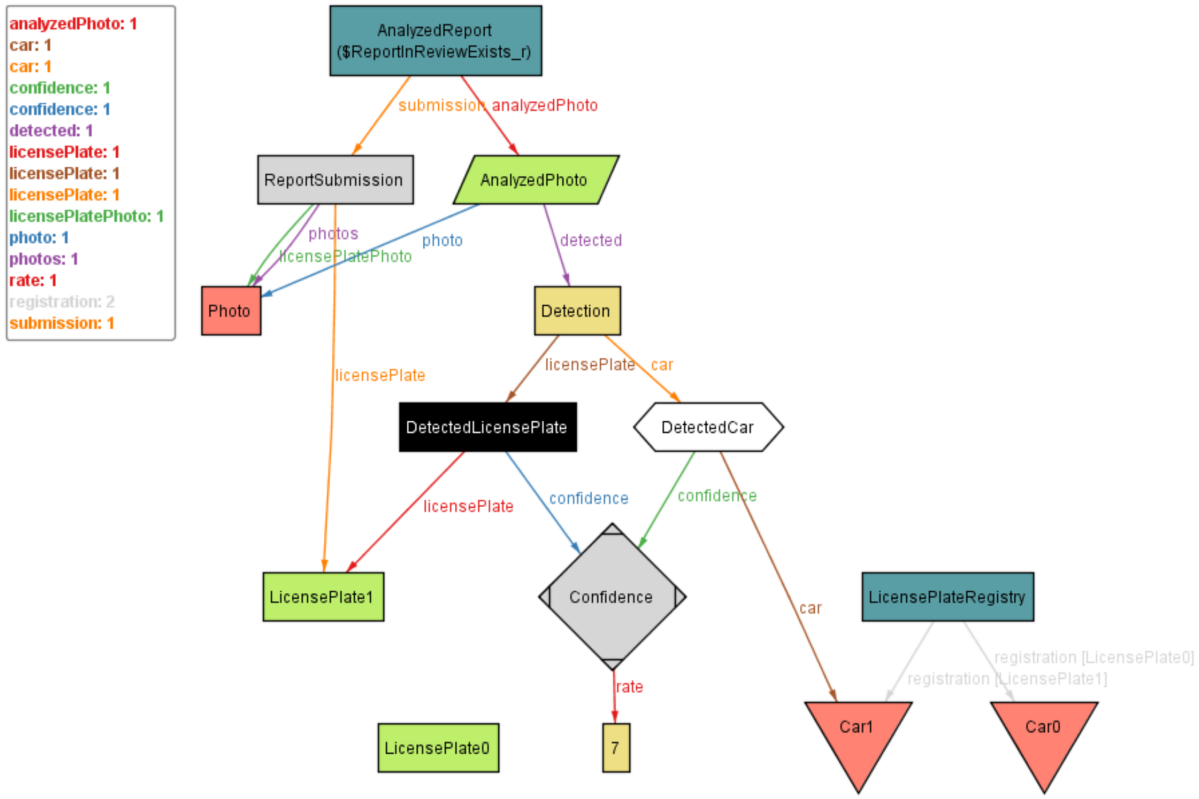

Figure 16: Alloy - World: Report in review.

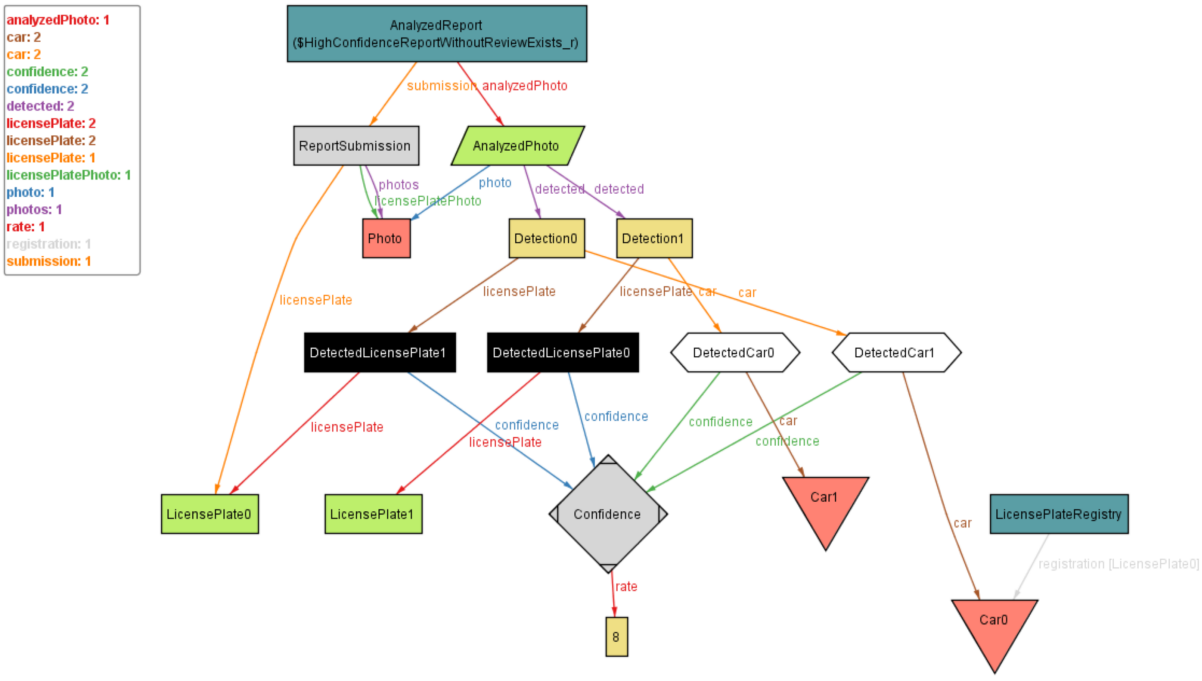

Figure 17: Alloy - World: High confidence report without a review.

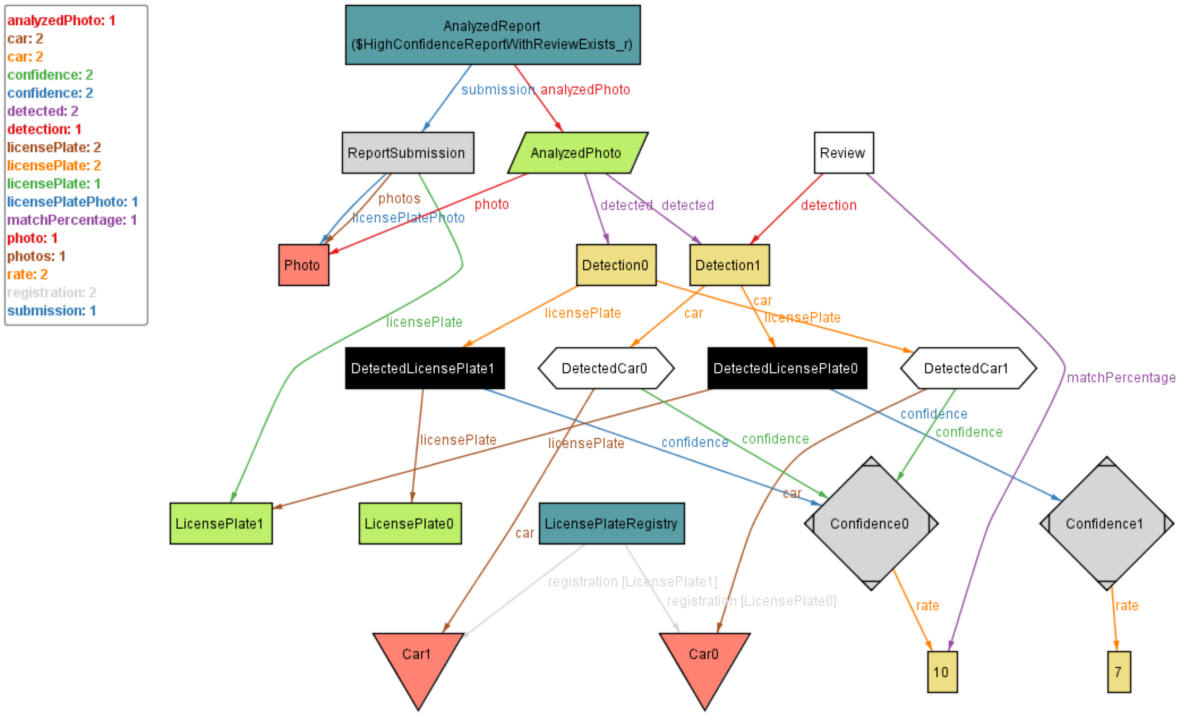

Figure 18: Alloy - World: High confidence report with a review.

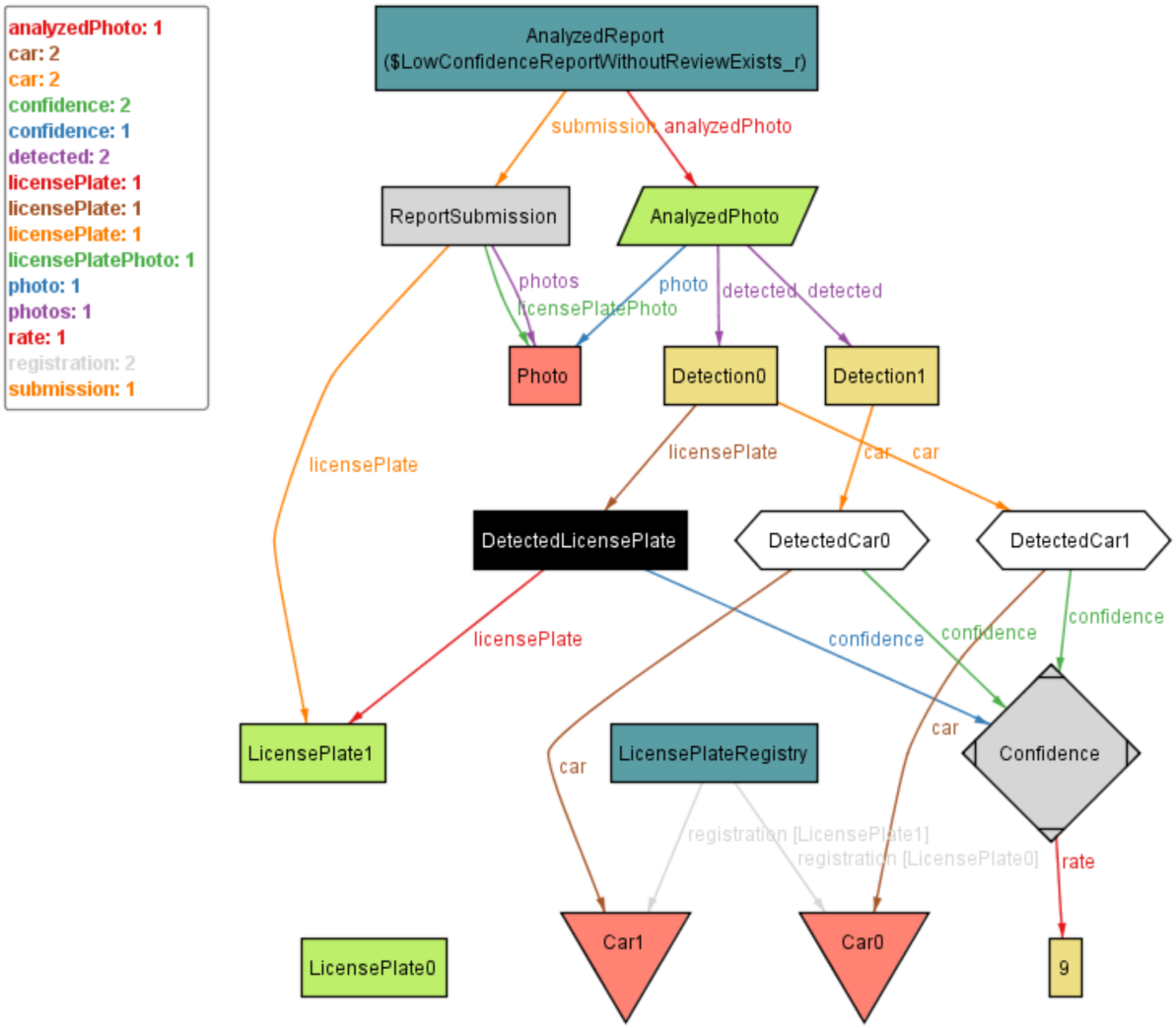

Figure 19: Alloy - World: Low confidence report without a review.

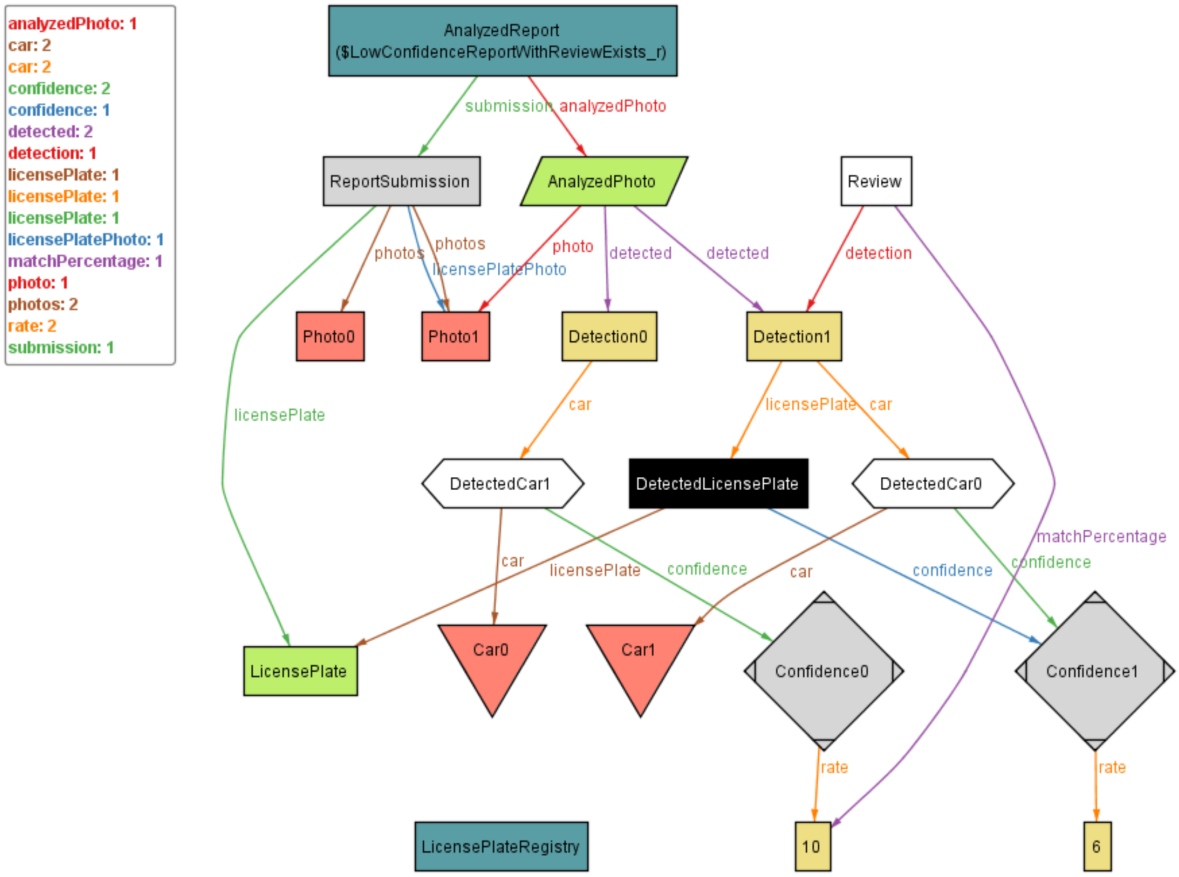

Figure 20: Alloy - World: Low confidence report with a review.

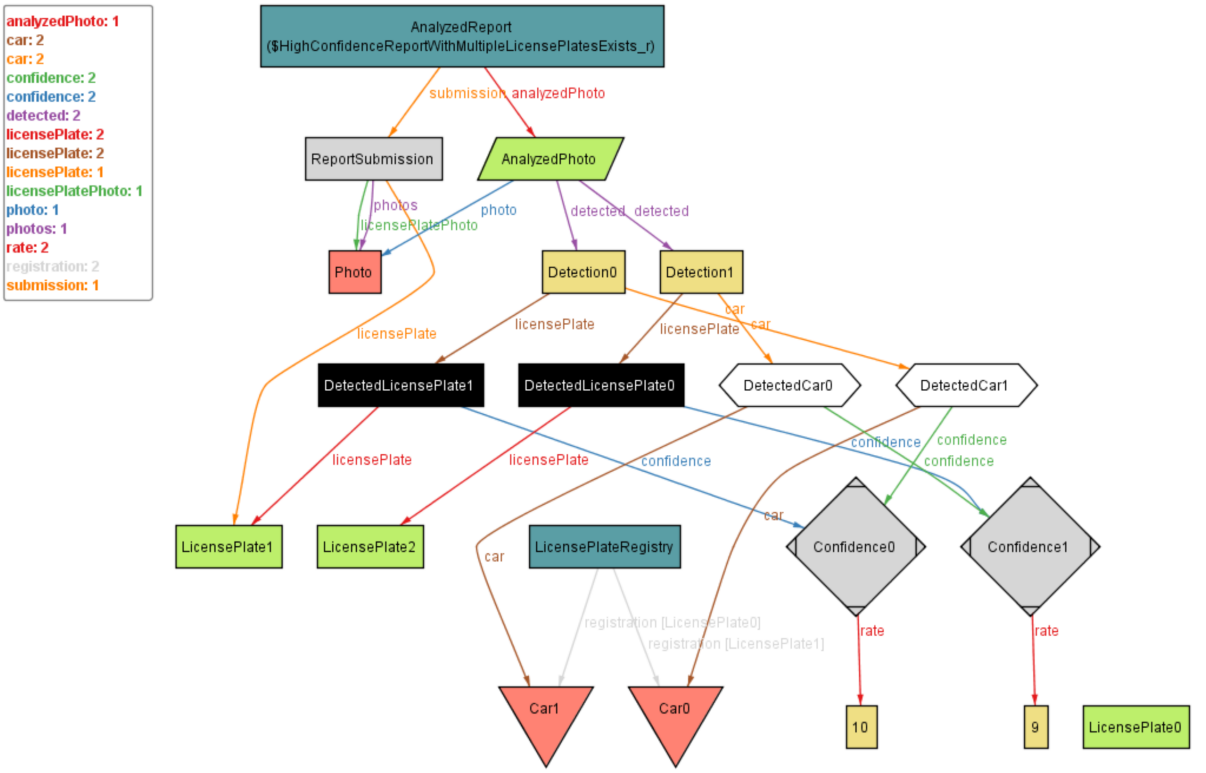

Figure 21: Alloy - World: High confidence report with multiple license plates.

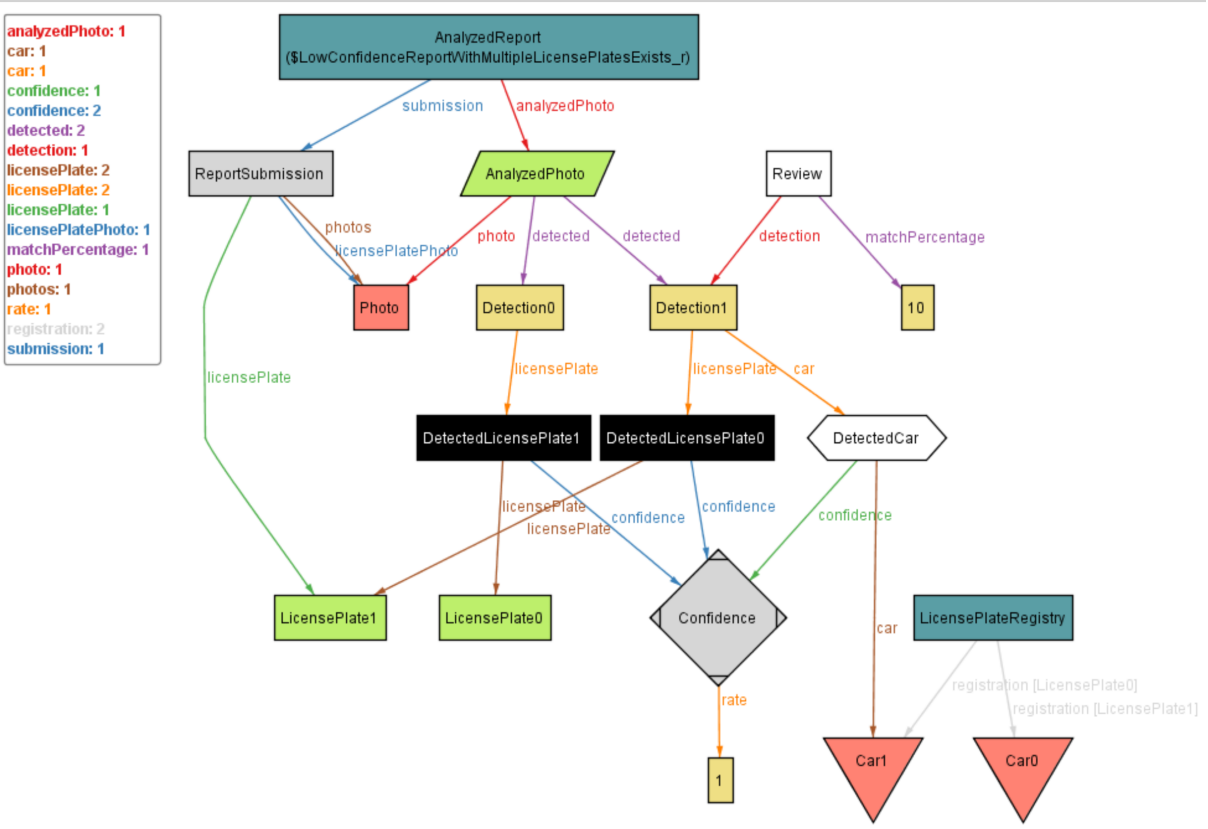

Figure 22: Alloy - World: Low confidence report with multiple license plates.

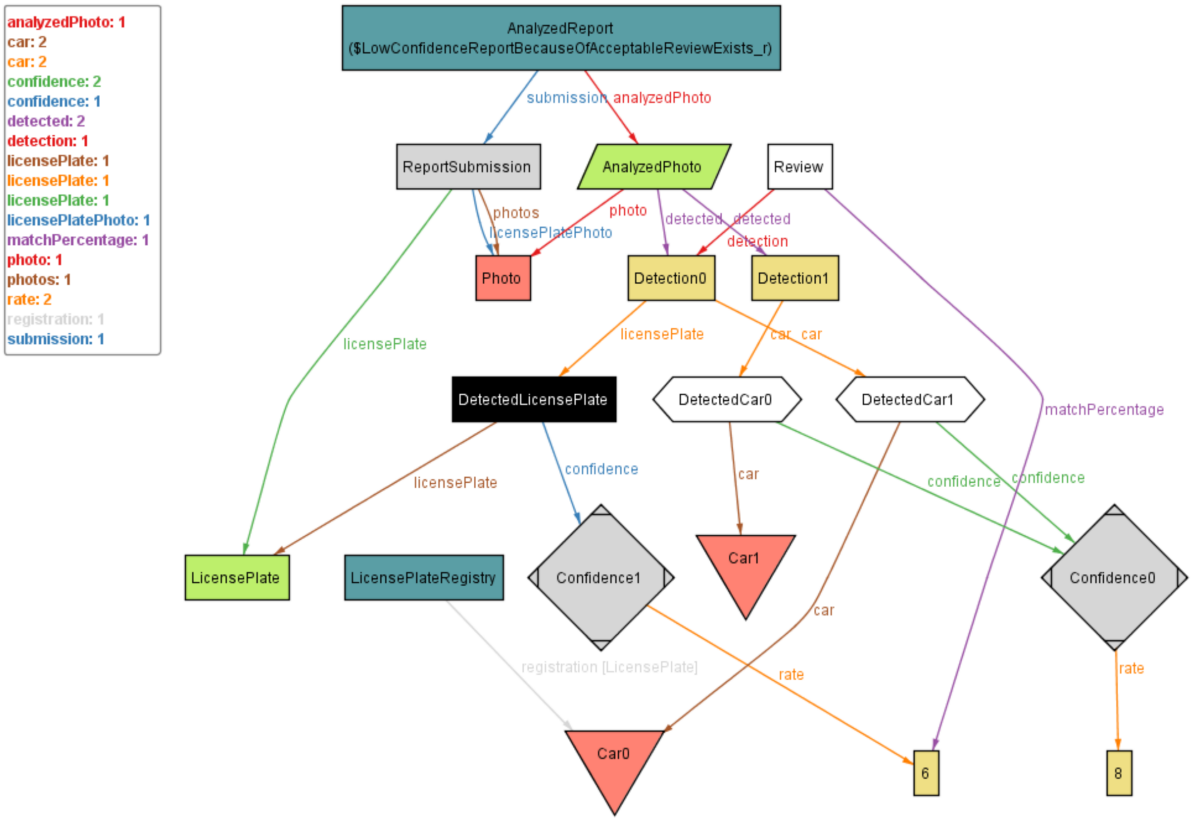

Figure 23: Alloy - World: Low confidence report because of an "Acceptable" review.

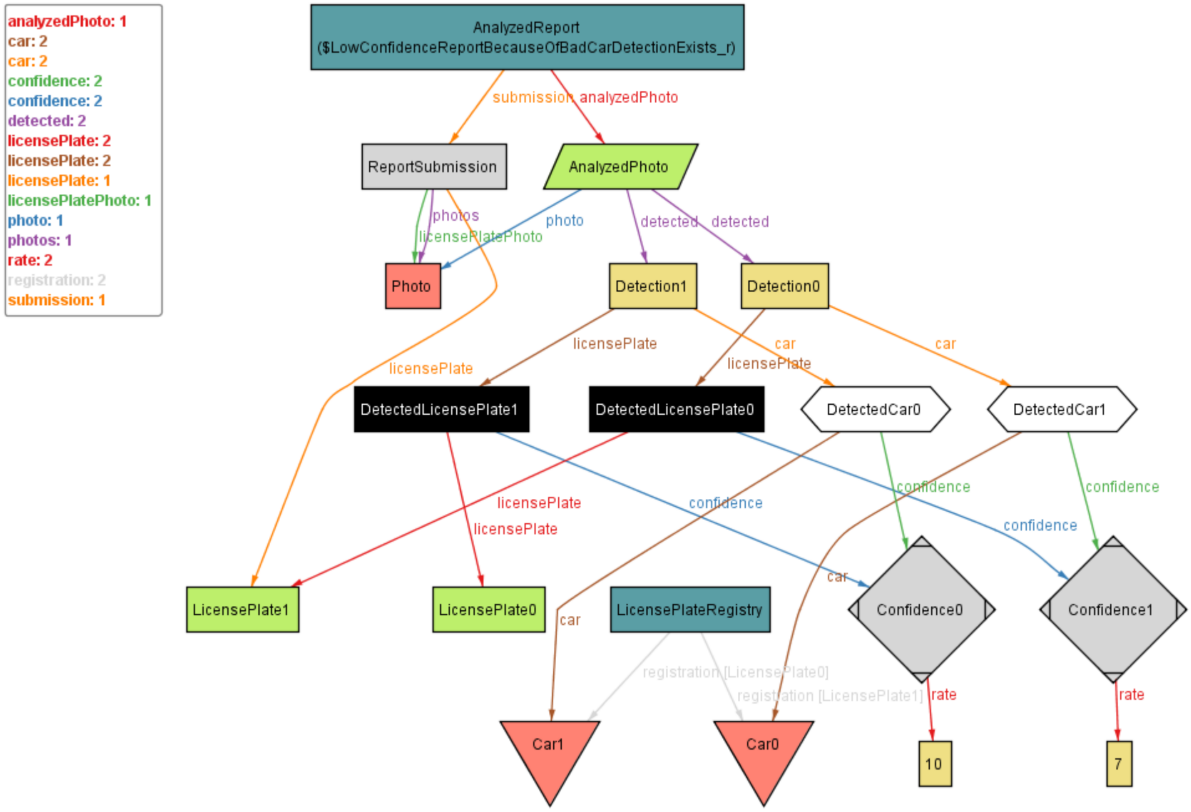

Figure 24: Alloy - World: Low confidence report because of a bad car detection.

Figure 25: Alloy - World: Low confidence report because there is no car registered under the detected license plate.

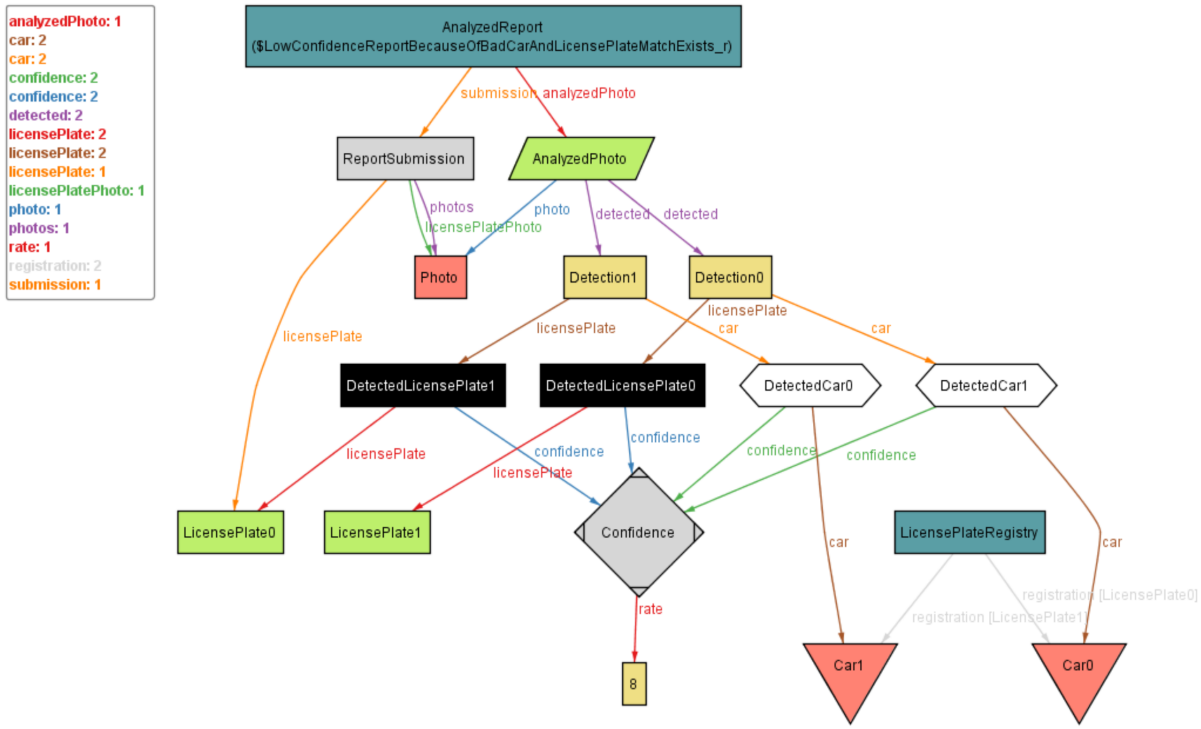

Figure 26: Alloy - World: Low confidence report because the detected car does not match the car registered under the detected license plate.

5 Effort Spent

Provide here information about how much effort each group member spent in working at this document. We would appreciate details here.

References

