504: Machine Learning meets Program Analysis

Assignment 1

Chenglong Wang

Data consistency problems in web applications

Problem When developing web-based user-interactive applications, keeping data-consistencies can be challenging. Data-consistency here means that difference occurrences of a same data should be consistent with each other: for example, given a web application with several occurrences of the current date (e.g. a date in the web page title, a date will appear in texts of a warning window and a date used in the internal computation of the program), all these different occurrences should be consistent – whenever a function modifies one of the data source, all of the occurrences should be modified correspondingly.

The challenge for a user (e.g. me) to keep data consistency in web application development is caused by the following facts.

- Web applications are interactive and the program status mutate when some event is triggered. As different event handlers are implemented in different functions and each handler need to modify different data occurrences in the application. As the code snippets relevant to the same data source are scattered in different locations of the program, it is challenging for programmers to keep these data consistent.
- When the data consistency is not guaranteed, identifying such inconsistency for user is challenging and they are likely to lead to latent bugs. This is caused by the fact that debugging interactive web applications requires generation of events and observing user interface modifications, as the observing process in this case requires the programmer to observe all web pages and it is hard to cover everywhere.

Problem analysis The key point of the problem may not be easily solved due to the following reasons.

- It is not easy to automatically identify which variables are supposed to be consistent with each other.
- Even if we can find out variables that are consistent with each other, analyzing whether they are consistent with each other is challenging. One particular reason is that most web applications are programmed using JavaScript and analyzing JavaScript programs is challenging.

Possible solutions Firstly, to identify variable occurrences that should be consistent each other, we can use natural language informations: 1) variable names can leave a hint, 2) texts in the web page around the data to be displayed can be used to infer the meaning of the data and 3) comments in the program can help identifying the meaning of an object. Once we have a list of potential variables, we can present them to the programmer and ask she to help confirm the list.

Secondly, when we obtain the variables, we can perform program analysis to check whether these variables are consistent with each other or generate test cases to check whether these variables are consistent with each other.

Automatic API migration

Problem When a new incompatible version of an API is released, users face to problem to modify their original code to use the new version of API. Such manual migration requires user energies to study the correspondence between the two version of API and it can be time consuming. In my experience of performing such migration process, I found an example with migration history and imitate it to migrate my program. Thus it is ideal to have a tool that can automatically identify API correspondence between the old and the new versions API based on such examples.

Why not solved? Existing method identifies such migration relations based only on this code migration history by evaluation AST subtree similarities between the old sample code and the new sample code. This approach 1) limits the migration relations that can be identified from the sample program and 2) evaluation of the similarity is based on empirical formula thus less convincing.

Possible solutions One way we can solve them is to learn source code correspondence with statistical methods and also adopting the API document. We can 1) firstly enumerate combinations of possible mappings based on the document, then 2) rank the combinations based on the development history (sample code) and 3) generate transformation rules to perform API migration.