Master's Thesis Description

Must analysis of collection elements

Yves Bonjour
Supervisor: Lucas Brutschy

Introduction

TouchDevelop [6], [5] is a novel programming language developed by Microsoft Research. It is geared towards development on mobile devices and also targets beginners with only little experience in programming. Under these circumstances it is important that the programming environment supports the user during the development process (e.g. by pointing out possible errors in the source code).

The TouchBoost project aims to improve the programming experience of TouchDevelop by statically analyzing the performance and correctness of scripts using static analysis.

The static analyses used in the TouchBoost project are implemented with Sample (Static Analyzer of Multiple Programming LanguagEs) [3], [4]. Sample is a generic static analyzer based on abstract interpretation [1], [2]. It has been developed at the Chair of Programming Methodology during the last three years. The analyzer is designed in a generic fashion which allows extensions for new domains and programming languages.

The TouchDevelop standard library implements several types of collections like lists or maps. For example the songs stored on a mobile device are represented as a collection of Song objects. Since the number of elements in a collection is potentially unbounded, we must abstract away from individual elements in our static analysis. A simple abstraction is to represent all elements in the collection by a single summary node, which over-approximates the set of possible elements.

In TouchDevelop, this approach often does not provide enough information to prove the desired property. The reason for this is, that we often have no static information about the collections involved in a script (e.g. when analyzing a library method that has a collection as a parameter). Since an over-approximation in this case always represents all possible elements it is not very useful.

Example

To illustrate this shortcoming we use the TouchDevelop script printed in listing 1 as an example. This code shows a publicly available action that prints the name of the US president based on a map that is passed into the action as an argument. It maps countries to their current presidents. One possible entry could be [$"USA"$ $\rightarrow$ "Barack Obama"].

```
1 action print_presidents(
2         presidents:String Map)
3 do
4         if presidents->keys->contains("USA") then
5                 presidents->at("USA")->post_to_wall
6         else
7                 "unknown"->post_to_wall
```

Listing 1: Print presidents

In this example we want to prove that the collection access at line 3 is safe.

The current analysis uses an over-approximation of the collection elements as described before. Since the collection *presidents* is a parameter of the action we don't know anything about the collection's elements at the beginning. Therefore the over-approximation abstracts the collection's elements with a summary node that represents all possible elements.

Inside the *if*-branch at line 3 we know that the collection contains a key *USA* since the expression $presidents- > keys- > contains("USA")$ must evaluate to *true* in order to enter this branch. The over-approximation however can not gain any additional information from that knowledge because it already represents all possible elements.

To prove that the collection access at line 3 is safe we need to be able to determine whether any concrete collection that is abstracted by the approximation contains the key *USA*. Since an over-approximation of the collection elements abstracts collections that contain the key *USA* but also collections that do not contain it, we are not able to prove that the collection access at line 3 is safe.

Goal

The main goal of this Master's thesis is to design, implement and formalize a technique, that is able to represent a valid under-approximation of collection elements. It is also part of this thesis to adapt the abstract semantics of the TouchDevelop collection operators such that it uses the information gained from said under-approximation. It shall for example be possible to resolve membership queries or to prove the existence of a particular element.

The technique shall be implemented as an extension to the existing analysis framework Sample.

The technique should be generic such that all types of collections that are currently defined in the TouchDevelop standard library can be represented. This includes lists, maps and collections of complex objects like songs.

A very successful project will deliver a technique which improves the precision in real world examples, has a formalization which can be proven sound using the abstract interpretation framework and is implemented in a generic fashion.

Extensions

Records

TouchDevelop offers a records library which allows the user to define new types like Tables or Indexes. These types are special cases of collections and are more complex than the predefined collections in the standard library. Indexes for example allow a key to be composed of multiple elements.

A possible extension to the project is to be able to represent Tables and Indexes.

Strings

Strings in a TouchDevelop scripts could be represented as lists of characters. This might bring more precision when analyzing scripts that contain string manipulations. It would for example allow to answer queries for the i-th character in a string.

A possible extension to this thesis is to explore whether this improves the precision in real-world examples and to extend the technique such that it can handle strings.

Partitioning

A possible extension to the project is an efficient partitioning of the under- and over- approximating representation of the collection elements according to the use of the collections in the program.

For example, one could partition the representation according the foreach-Guards in the program.

Project Management

Project plan

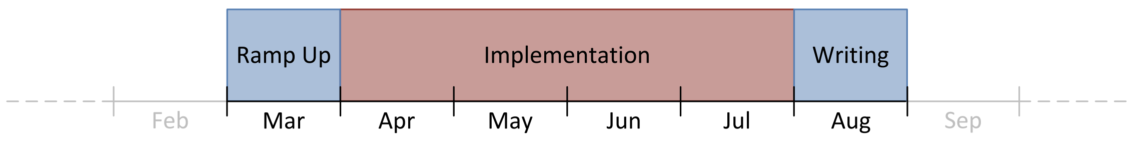

Figure 1: Coarse project plan

Project Start: March 1st 2013
Project End: September 1st 2013

As depicted in Figure 1 the project is divided into three phases. However those three phases are not strictly separated but may overlap.

Ramp Up Phase

This phase is used to set up the environment and to get familiar with the TouchDevelop library and Sample. Furthermore sample programs for which an under-approximation of collection elements could improve an analysis shall be collected. Also there should be an idea of how the problem could be solved.

Deliverables:

- Project description
- Initial presentation
- Set of sample programs

Start: March 4th 2013
End: April 11th 2013 (initial presentation)

Implementation Phase

During the implementation phase the collection abstraction is designed, implemented, evaluated and formalized. This is done in a series iterations.

In the middle of the implementation phase there will be a presentation about the ongoing work.

Start: April 12th 2013
End: August 2nd 2013

Deliverables:

- Source code
- Formal description of technique
- Evaluation results
- Intermediate presentation

Writing Phase

The writing phase is used to describe and present the implementation, formalization as well as the evaluation results.

Deliverables:

- Final report
- Final presentation

Start: August 8th 2013
End: September 1st 2013

Iterations

The length of an iteration during the implementation phase will be one week.

After each week the progress of the project will be reviewed and the goals and tasks for the next week will be defined. This task will be done in collaboration with the project supervisor. Therefore weekly meetings are hold.

References

[1] P. Cousot and R. Cousot. "Abstract Interpretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". In: *In Proceedings of POPL '77.* ACM Press, 1977.

[2] P. Cousot and R. Cousot. "Systematic Design of Program Analysis Frameworks". In: *In Proceedings of POPL '79.* ACM Press, 1979.

[3] P. Ferrara. "Static Type Analysis of Pattern Matching by Abstract Interpretation". In: *InProceedings of FMOODS '10.* 2010, pp. 186–200.

[4] P. Ferrara, R. Fuchs, and U. Juhasz. "TVAL+: TVLA and Value Analyses Together". In: *In Proceedings of SEFM '12.* 2012, pp. 63–77.

[5] N. Horspool et al. "TouchDevelop – Programming on a Phone. Version 1.1 for TouchDevelop 2.8". Microsoft Research, May 2012.

[6] Microsoft Research. *TouchDevelop.* URL: `http://research.microsoft.com/en-us/projects/touchdevelop/` (visited on 03/08/2013).