Using Consensus to Automatically Infer Post-conditions

Bowling Green State University
Department of Computer Science

Jingyi Su jsu@bgsu.edu
Mohd Arafat marafat@bgsu.edu
Robert Dyer rdyer@bgsu.edu

Problem and Motivation

MOTIVATION

Using APIs without formal specifications can put software systems at safety and security risks. Formal specifications help in producing cost effective, secure and reliable systems.

PROBLEM

- Post-conditions not widely available for common APIs
- Writing specifications is:
  - Effort intensive
  - Time consuming
  - Cost intensive
  - Difficult!

- **Automation is needed!**

Related

- *Nguyen et al. [FSE'14]*
  - Mined API pre-conditions using consensus approach
- *Chang et al. [VMCAI'11]*
  - Mined conditions using dependence graphs
- *Kremenek et al. [OSDI'06]*
  - Used factor graph to infer specification from programs
- *Ammons et al. [POPL'02]*
  - Mined formal specifications using machine learning
- *Ernst et al. [ICSE'99]*
  - Discovered invariants from execution traces using dynamic techniques

Proposed Approach

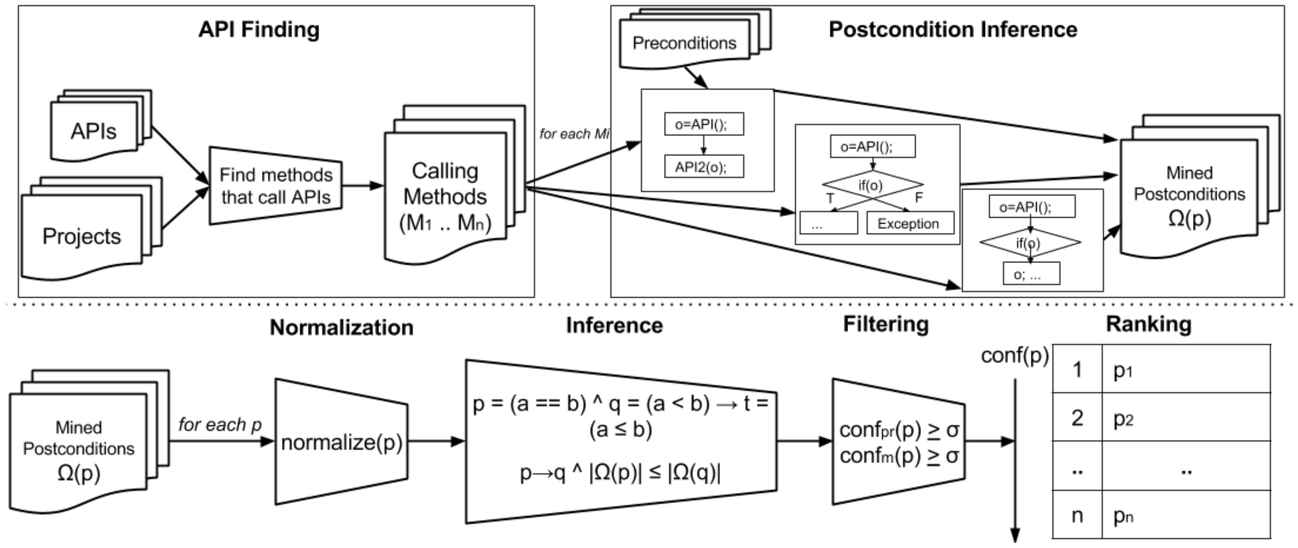
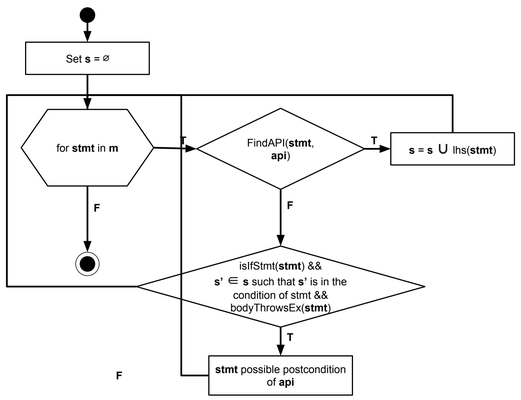


Image based on Nguyen et al. [FSE'14]

- Use ultra-large-scale open-source software repositories to automatically infer postconditions for existing APIs
- Key Insight: an API's original postcondition(s) occur more frequently than project- or code-specific postconditions
- Boa's existing infrastructure coupled with above approach addresses all the problems
- Steps involved:
  - API Finding
  - (new) Post-condition inference
  - Normalization
  - Inference
  - Filtering
  - Ranking

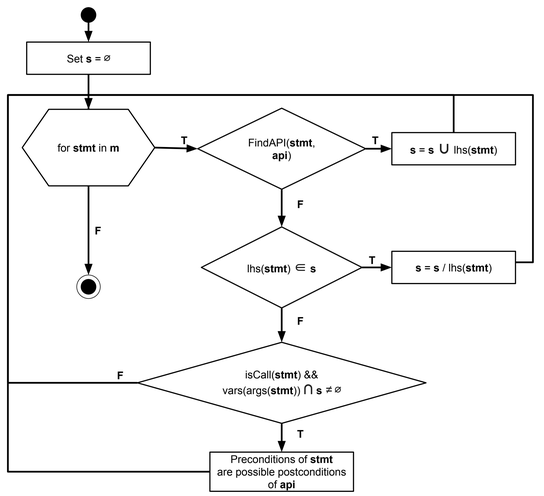
Approach 1: Finding throws clauses

```
o = API(...);
... // code that does not modify o
if (o > 5) {
  ...
  throw new Exception();
}
```



Approach 2: Using Preconditions

```
o = API(...);
... // code that does not modify o
anotherMethodWithPreconditions(o);
```



Evaluation Plan

- Evaluation set-up
  - Ultra-large GitHub data set in the Boa infrastructure
  - Use Java Modeling Language's hand-coded post-conditions as Ground-Truth

- A mined post-condition is considered correct in relation to a ground truth if:
  - it exactly matches with one of the API's post-conditions; or
  - it is not present in ground-truth but manually verified to be correct; or
  - it is not-yet defined but semantically equivalent to an existing post-condition; or
  - it is not-yet defined but implied by a post-condition.