**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

**Evaluation Plan**

- 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

- 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.