Deep Typechecking and Refactoring

Zachary Tatlock, Chris Tucker, David Shuffleton, Ranjit Jhala, and Sorin Lerner

Computer Science and Engineering, UCSD

Communicating with Databases

Database interaction is crucial for many programs, especially web applications. Unfortunately, the Database / Programming Language *Impedance Mismatch* makes this interaction difficult.

A prevalent solution is to embed query strings directly in the application. This approach is *efficient* and *flexible* but **unsafe.**

The Problem :

Database query fragments are scattered throughout the program.

These fragments are **opaque** to the compiler.

Errors from DB interaction aren't caught until runtime!

```
String getText(String id, Link link) {
  String qStr;
  Query q;

  qStr  = "SELECT w FROM Weblog w ";
  qStr += "WHERE w.id = ?1 ";
  qStr += "AND w.link.id = ?2";

  q = createQuery(qStr);
  q.setParam(1, id);
  q.setParam(2, link.id);

  Weblog w = (Weblog) q.execQuery();

  return w.text;
}
```

A

B

C

To ensure **safety**, we must guarantee:

- **A.** All query parameters are set
- **B.** All parameters are set to the correct type
- **C.** Query results are safely used (correctly downcast)

Deep Typechecking ensures that these properties hold.

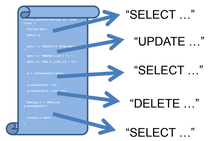
Deep Refactoring extends this analysis to enable common software engineering tasks.

Deep Typechecking

Goal:

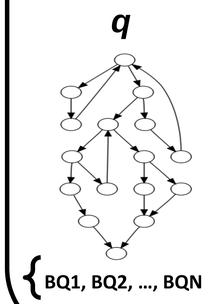
Ensure there are no bugs from DB interaction.

Step 1: String Analysis



For each query object **q**, compute the query strings **q** may represent.

Step 2: Bound Query Analysis ★



Track how each query object is modified, right up to the point it is executed.

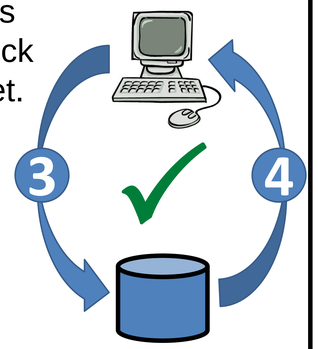
Yields a set of bindings from a parameter to the types of data it has been set to.

Step 3: Check Parameters

Use the set of bound queries and the query strings to check that all params are safely set.

Step 4: Check Results

Track query result to each use and ensure safe downcast.



Conclusions:

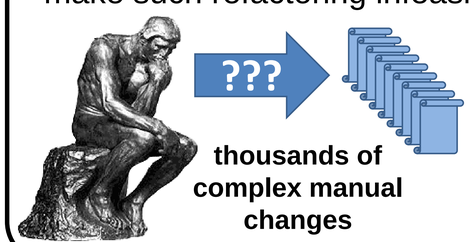
1. Deep Typechecking ensures safety
- No silent failures

2. Robust in the face of imprecision
- Param and result checks independent

3. Effective in practice
- Analyze 100K lines of industrial code in 40s
- Able to prove 85% of exec sites safe

Deep Refactoring

Renaming classes and fields is a basic and frequent software engineering task. Unfortunately, string based queries can make such refactoring infeasible.



Example: Weblog.id → Weblog.name

```
qStr  = "SELECT w FROM Weblog w ";
...
qStr += "WHERE w.id = ?1 ";   ?
...
qStr += "AND w.link.id = ?2"; ?
```

Bound Query Analysis

```
qStr  = "SELECT w FROM Weblog w ";
...
qStr += "WHERE w.name = ?1 ";   ✓
...
qStr += "AND w.link.id = ?2"; ✗
```

Bound Query Analysis is extended to refactor full query strings. Changes are propagated back to the source.