Current Development of CPAchecker

Philipp Wendler

CPA19
2019-10-02

Issue Tracker

- Discussion at CPA17:
  - Move to GitLab in hope of higher participation
  - Keep private for now
- GitLab is indeed nicely used (3 issues/week, with Trac < 2)
- Access given on request for developers and prolific reporters
- Vote: Make issue tracker public?
  - Will also make old issues public
  - Users can file issues
  - Easier communication with reports on mailing list

Issue Tracker

- Not only for bugs, use it freely how you want:
  - Project ideas
  - Milestones, issue boards
- Current label concept:
  - Affected component (Component: …)
  - CPAchecker feature (witness, refinement, preprocessing, …)
  - C feature (aliasing, union, floats, …)
  - Kind (unsound, imprecise, performance, usability, …)

Java 11

- Suggested switch to Java 11 on mailing list
- No complaints
- Plan:
  - Release with LTL model checking and Java 8 these days
  - Switch to Java 11 directly afterwards

New Features in Java 11

- Easier data-structure construction:
  - `List.of(…) / Set.of(…) / Map.of(…)`
  - `List.copyOf(…)` etc.
  - Please keep using `Immutable{List,Set,Map}.{of,copyOf}(…)` instead!
  - Immutability in type, deterministic order, consistency

New Features in Java 11

- Type inference of local variables:
  - `var foo = bar;`
  - Do not use to save effort while writing code!
  - Use if code readability is increased.
  - Code is read more often than written!
  - Example:

```
for (var entry : foo.getVariableTypes().entrySet()) {
  CVariableDeclaration key = entry.getKey();
  CType value = entry.getValue();
  …
}
```

New CI Check: Refaster

- Google project based on Error-Prone: https://errorprone.info/docs/refaster
- Rule-based code refactorings
  - Write templates and refaster finds instances
  - Write suggested replacement and refaster generates patch
  - Rules are regular Java code!

Refaster Example (1)

```
class ImmutableListCopyOf<T> {

  @BeforeTemplate
  List<T> listCopyOf(Collection<T> c) {
    return List.copyOf(c);
  }

  @AfterTemplate
  ImmutableList<T> immutableListCopyOf(Collection<T> c) {
    return ImmutableList.copyOf(c);
  }
}
```

Refaster Example (2)

```
class StringIsNullOrEmpty {

  @BeforeTemplate
  boolean isNullOrEmpty(String s) {
    return s == null || s.isEmpty();
  }

  @AfterTemplate
  @AlsoNegation
  boolean stringsIsNullOrEmpty(String s) {
    return Strings.isNullOrEmpty(s);
  }
}
```

Also finds
`s != null && !s.isEmpty()`

Refaster Example (3)

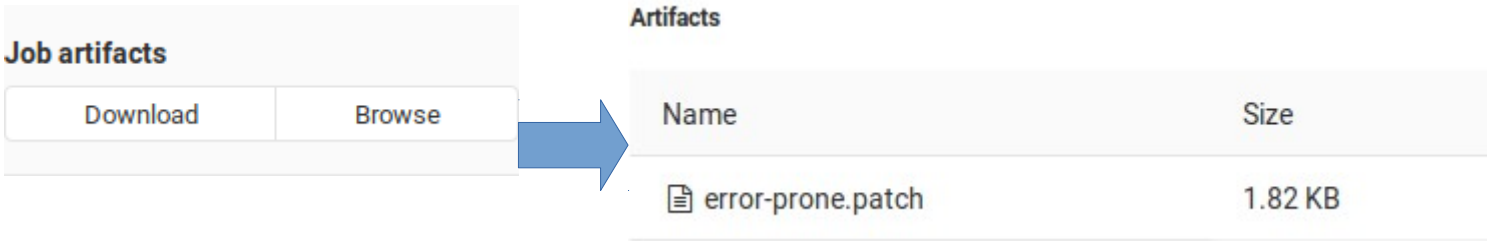
```
@Placeholder
abstract void doAfterAdd(@MayOptionallyUse E element);

@BeforeTemplate
void ifNotContainsThenAdd(Set<E> set, E element) {
  if (!set.contains(element)) {
    set.add(element);
    doAfterAdd(element);
  }
}

@AfterTemplate
void ifAdd(Set<E> set, E element) {
  if (set.add(element)) {
    doAfterAdd(element);
  }
}
```

New CI Check: Refaster

- Usage needs to be polished
- But integrated in our CI, just download patch from GitLab:




```
patch -p0 < error-prone.patch
```

- Apply google-java-format afterwards (`git add; ant format-diff`)
- Repository with our rules: https://gitlab.com/sosy-lab/software/refaster
- Suggestions welcome!

Machine-Readable Statistics (#356)

- Statistics as JSON
- Work in progress
  - Open question: structure of JSON
    (statistic values might be present more than once)
- Will be implemented incrementally
  - Central framework
  - Individual statistics of each component

Machine-Readable Statistics (#356)

- Preparation can start already!
- Use `org.sosy_lab.cpachecker.util.statistics.Stat*`
  - Couples statistics value with human-readable title
    (and in future machine-readable key)
  - `StatTimer` as replacement for `Timer`
  - `StatInt` as accumulator with min/max/avg
- Output using `StatisticsWriter.put(AbstractStatValue)`
- Will allow StatisticsWriter to output machine-readable format, too