jMoped
A Checker for Java Programs

Dejvuth Suwimonteerabuth

Technische Universität München

Joint work with Felix Berger, Stefan Schwoon, and Javier Esparza
jMoped: an Eclipse plug-in

jMoped performs a reachability analysis to check the reduced Java method.
jMoped: an Eclipse plug-in

Generates coverage information from model-checking results.

![Diagram showing jMoped generating coverage information from Java source code.](image-url)
jMoped: an Eclipse plug-in

Tests for common Java errors, i.e. assertion violations, null-pointer exceptions, array bound violations.
jMoped: an Eclipse plug-in

Generates JUnit test cases or call traces with concrete inputs.
jMoped: an Eclipse plug-in

Searches backwards from uncovered instructions.

ranges \rightarrow J\text{ava source} \rightarrow j\text{Moped} \rightarrow \begin{align*}
\text{not covered} & \quad \text{mark with } \bullet \\
\text{covered} & \quad \text{mark with } \bullet \\
\text{exception error} & \quad \text{mark with } \bullet \uparrow [j]
\end{align*}

\begin{align*}
\text{not covered} & \quad \text{backward analysis} \\
\text{covered} & \quad \text{create J\text{Unit test case and/or create call trace}}
\end{align*}
Overview of the Architecture

- Remopla
  - moped
    - reachable states
Overview of the Architecture

Java source → javac → bytecode

→ Remopla → moped → reachable states
Overview of the Architecture

Java source

javac

bytecode

translator

Remopla

moped

reachable states
Overview of the Architecture

1. Java source
2. javac
3. bytecode
4. translator
5. Remopla
6. moped
7. reachable states

Diagram:
- Java source -> javac -> bytecode
- Bytecode -> translator
- Translator -> Remopla
- Remopla -> reachable states
Overview of the Architecture

- **plug-in**
  - Java source
  - **javadoc**
    - bytecode
  - **translator**
    - **Remopla**
      - reachable states

- coverage result
Supported Java Features

- Assignments
- Control statements
- Method calls and recursions
- Exceptions
- Strings (very limited)
- Multi-dimensional arrays
- Object-oriented programming
  - Inheritance
  - Abstraction
  - Polymorphism
Limitations

- No float or negative values
- No multi-threading
- State space is rebuilt for each analysis

Performance limitations

- Analysis: jMoped is too slow for heaps bigger than 64 blocks
- Translation: A class in the Java library often calls many other classes → very big program!
Demonstrations

More demos ...
Conclusion

- Symbolic testing: uses a BDD-based model checker for testing a large set of inputs.
- Generates coverage information and find some common errors.
- User-friendly interface, model checker is hidden.
- Can be used as a complement to JUnit.
- Supports backward analysis.

http://www7.in.tum.de/tools/jmoped