A Program Logic for Bytecode[1]

Fabian Bannwart    Peter Müller

Presented by Moritz Hoffmann

April 15, 2013
Objective

A sound and complete Hoare-style logic to apply Proof-Carrying Code on bytecode.
Motivation

- Intermediate languages are part of standardized execution environments, i.e. JVM and .NET.
- Formal reasoning on source level
- Improve and speed up JIT
Proof-carrying code

- Translate verified source code to verified bytecode
- Annotate intermediate language with proofs
- Efficient run-time verification of proof carrying code

Problem
- Code is compiled to intermediate language
- Source proof must also be transformed

Goal
Develop proof-transforming compiler
Bytecode Language

Bytecode language VM$_K$

- Used to model classes, methods and instructions
- No exception handling
- Programs are well typed
- Object Store models heap
- Stack
- Similar to JVM and CLI bytecode instructions

Hoare-style rules for every included instruction
Program Logic

[The program logic] allows to formally verify that implementations satisfy interface specifications given as pre- and postconditions.

Method specification \( \{P\} \text{ comp } \{Q\} \)

Instruction specification \( \mathcal{A} \vdash \{E_i\} i : l_i \)

Method Sequence of instruction specifications
\[ \forall i \in \{0, \ldots, |\text{body}(T@m)| - 1\} : (\mathcal{A} \vdash \{E_i\} i : l_i) \]

\[ \forall i \in \{0, \ldots, |\text{body}(T@m)| - 1\} : (\mathcal{A} \vdash \{E_i\} i : l_i) \]
\[ \mathcal{A} \vdash \{E_o\} \text{ body}(T@m) \{E|\text{body}(T@m)|-1\} \]
Method Specification

- Method specification: \{P\} \text{comp} \{Q\}
- \text{comp} is a method implementation $T@m$ of a virtual interface $T : m$
- Support for virtual methods
- Contains language independent rules to connect method specifications to programming logic
Instruction Specification

- Instruction specification: \( \{ E_i \} \ i : l_i \)
- \( \{ E_i \} \) is the local weakest precondition
- \textit{shift} and \textit{unshift} model stack operations.

<table>
<thead>
<tr>
<th>( l_i )</th>
<th>( \text{wp}_p^1 (l_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>pushc ( v )</td>
<td>( \text{unshift}(E_{i+1}[v/s(0)]) )</td>
</tr>
<tr>
<td>pop ( \times )</td>
<td>( (\text{shift}(E_{i+1}))[s(0)/\times] )</td>
</tr>
<tr>
<td>binop_{\text{op}}</td>
<td>( (\text{shift}(E_{i+1}))[s(1) \text{ op } s(0)]/s(1)] )</td>
</tr>
</tbody>
</table>

- Other operations: pushv, goto, brtrue, checkcast, newobj, getfield, putfield, return
Application

\{ p = P \} Math : abs

\{ (P \geq 0 \Rightarrow result = P) \land (P < 0 \Rightarrow result = -P) \}

Since Math : abs is defined in a class without subclasses, we can apply the following rule:

\[
\begin{align*}
\{ p = P \land \tau(this) = Math \land this \neq null \} & \quad body(Math@abs) \{ Q \} \\
\{ p = P \land \tau(this) = Math \} & \quad Math@abs \{ Q \}
\end{align*}
\]
Impact

- Development of proof-transforming compiler producing proof-carrying code
- Foundation to understanding complication of `break` and `try/catch/finally` clauses

Remaining issue

- Only one method parameter $p$ is covered by the logic.
- Logic does not handle type checking
Proof Math : abs

\[ p = P \land \tau(\text{this}) = \text{Main} \land \text{this} \neq \text{null} \]
0 : pushv \( p \)

\[ (s(0) < 0 \Rightarrow P < 0) \land (s(0) \geq 0 \Rightarrow P \geq 0) \land p = P \]
1 : pushc 0

\[ (s(1) < s(0) \Rightarrow P < 0) \land (s(1) \geq s(0) \Rightarrow P \geq 0) \land p = P \]
2 : binop \( \geq \)

\[ (s(0) < 0 \Rightarrow P < 0) \land (s(0) \geq 0 \Rightarrow P \geq 0) \land p = P \]
3 : brtrue 8

\[ P < 0 \land p = P \]
4 : pushc 0

\[ P < 0 \land s(0) - p = -P \]
5 : pushv \( p \)

\[ P < 0 \land s(1) - s(0) = -P \]
6 : binop \( \_ \)

\[ P < 0 \land s(0) = -P \]
7 : goto 9

\[ P \geq 0 \land p = P \]
8 : pushv \( p \)

\[ (P \geq 0 \Rightarrow s(0) = P) \land (P < 0 \Rightarrow s(0) = -P) \]
9 : pop result

\[ (P \geq 0 \Rightarrow \text{result} = P) \land (P < 0 \Rightarrow \text{result} = -P) \]
10 : return
Transformation of Source Proofs

\[ S \left( \frac{T_{\{e \land P\}} S \{P\}}{\{P\} \text{ while } (e) S \{\neg e \land P\}} \right) = \]

\[
\begin{align*}
\{P\}l_1 : \\
\{e \land P\}l_2 : \\
\{P\}l_3 : \\
\{\text{shift}(P) \land s(0) = e\}l_4 : \\
\{P \land \neg e\}
\end{align*}
\]

\[
\text{goto } l_3 \\
S \left( \frac{T_{\{e \land P\}} S \{P\}}{S_E(P, e)} \right) \\
\text{brtrue } l_2
\]
For Further Reading I