Formal Methods for Java

Lecture 2: Operational Semantics

Jochen Hoenicke

Software Engineering
Albert-Ludwigs-University Freiburg

April 19, 2007
Semantics for Java

The Java Language Specification (JLS) 3rd edition gives semantics for Java

- The document has 684 pages.
- 118 pages to define semantics of expression.
- 42 pages to define semantics of method invocation.

Semantics are only defined by prosa text.
Can we do it formally?
Idea: define transition system for Java

**Definition (Transition System)**

A transition system \((TS)\) is a structure \(TS = (Q, \text{Act}, \rightarrow)\), where

- \(Q\) is a set of states,
- \(\text{Act}\) a set of actions,
- \(\rightarrow \subseteq Q \times \text{Act} \times Q\) the transition relation.

- \(Q\) reflects the current dynamic state (heap and local variables).
- \(\text{Act}\) is the executed code.
What is the state after executing this code?

```java
List mylist = new LinkedList();
mylist.add(new Integer(1));
```
The state of a Java program gives valuations local and global (heap) variables.

- $Q = \text{Heap} \times \text{Local}$
- $\text{Heap} = \text{Address} \rightarrow \text{Class} \times \text{seq Value}$
- $\text{Local} = \text{Identifier} \rightarrow \text{Value}$
- $\text{Value} = \mathbb{Z}$, $\text{Address} \subseteq \mathbb{Z}$

A state is denoted as $(\text{heap}, \text{lcl})$, where $\text{heap} : \text{Heap}$ and $\text{lcl} : \text{Local}$.
An action of a Java Program is either
- the evaluation of an expression $e$ to a value $v$, denoted as $e \triangleright v$, or
- a Java statement, or
- a Java code block.

Note that expressions with side-effects can modify the current state.
Example: Actions of a Java Program

Post-increment expression:

\[(heap, lcl \cup \{x \mapsto 5\}) \xrightarrow{x++ \triangleright 5} (heap, lcl \cup \{x \mapsto 6\})\]

Pre-increment expression:

\[(heap, lcl \cup \{x \mapsto 5\}) \xrightarrow{++x \triangleright 6} (heap, lcl \cup \{x \mapsto 6\})\]

Assignment expression:

\[(heap, lcl \cup \{x \mapsto 5\}) \xrightarrow{x=x*2 \triangleright 10} (heap, lcl \cup \{x \mapsto 10\})\]

Assignment statement:

\[(heap, lcl \cup \{x \mapsto 5\}) \xrightarrow{x=x*2} (heap, lcl \cup \{x \mapsto 10\})\]
The last slide listed some examples for transitions. Define rules when a transition is valid.

**Definition (Inference Rules)**

A rule of inference

$$
\frac{F_1 \ldots F_n}{G}, \text{ where } \ldots
$$

is a **decidable** relation between formulae. The formulae $F_1,\ldots,F_n$ are called the **premises** of the rule and $G$ is called the conclusion.

If $n = 0$ the rule is called an **axiom schema**. In this case the bar may be omitted.

The intuition of a rule is that if all premises hold, the conclusion also holds.
axiom for evaluating local variables:

\[(heap, lcl) \xrightarrow{x \mapsto lcl(x)} (heap, lcl)\]

rule for field access:

\[\begin{align*}
(\text{heap}, \text{lcl}) \xrightarrow{\text{e}\triangleright v} (\text{heap}', \text{lcl}') \\
(\text{heap}, \text{lcl}) \xrightarrow{\text{e.fld}\triangleright \text{heap}'(v)(\text{idx})} (\text{heap}', \text{lcl}')
\end{align*}\]

where \(\text{idx}\) is the index of the field \(\text{fld}\) in the object \(\text{heap}'(v)\)

rule for assignment to local:

\[\begin{align*}
(\text{heap}, \text{lcl}) \xrightarrow{\text{e}\triangleright v} (\text{heap}', \text{lcl}') \\
(\text{heap}, \text{lcl}) \xrightarrow{x = \text{e}\triangleright v} (\text{heap}', \text{lcl}' \oplus \{x \mapsto v\})
\end{align*}\]
Rules for Java expressions (2)

axiom for evaluating a constant expression \( c \):

\[
(\text{heap}, lcl) \xrightarrow{c > c} (\text{heap}, lcl)
\]

axiom for multiplication (similar for other binary operators)

\[
\begin{align*}
(\text{heap}_1, lcl_1) & \xrightarrow{e_1 > v_1} (\text{heap}_2, lcl_2) \\
(\text{heap}_2, lcl_2) & \xrightarrow{e_2 > v_2} (\text{heap}_3, lcl_3) \\
\hline
(\text{heap}_1, lcl_1) & \xrightarrow{e_1 \ast e_2 > (v_1 \cdot v_2 \mod 2^{32})} (\text{heap}_3, lcl_3)
\end{align*}
\]
A derivation for $x = x * 2$

\[
(\text{heap, lcl } \cup \{x \mapsto 5\}) \xrightarrow{x \triangleright 5} (\text{heap, lcl } \cup \{x \mapsto 5\}) \xrightarrow{2 \triangleright 2} (\text{heap, lcl } \cup \{x \mapsto 5\}) \xrightarrow{x \triangleright 10} (\text{heap, lcl } \cup \{x \mapsto 10\})
\]