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.

How can we give the semantics formally?
Need a mathematical model for computations.
Idea: define transition system for Java

**Definition (Transition System)**

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

- $Q$ is a set of states,
- $Act$ is a set of actions,
- $\rightarrow \subseteq Q \times Act \times Q$ the transition relation.

- $Q$ reflects the current dynamic state (heap and local variables).
- $Act$ is the executed code.
Example: State of a Java Program

What is the state after executing this code?

```java
List mylist = new LinkedList();
mylist.add(new Integer(1));
```

![Heap diagram showing the state after execution of the code.](diagram.png)
The state of a Java program gives valuations to local and global (heap) variables.

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

A state is denoted as \((heap, lcl)\), where \(heap : Heap\) and \(lcl : Local\).
Actions of a Java Program

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{\text{\(x++\)}} (heap, lcl \cup \{ x \mapsto 6 \})\]

Pre-increment expression:

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

Assignment expression:

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

Assignment statement:

\[(heap, lcl \cup \{ x \mapsto 5 \}) \xrightarrow{\text{\(x=x*2;\)}} (heap, lcl \cup \{ x \mapsto 10 \})\]
The last slide listed some examples for transitions. 
We now 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:

\[(heap, lcl) \xrightarrow{e \triangleright v} (heap', lcl')\]

\[(heap, lcl) \xrightarrow{e.fld \triangleright heap'(v)(idx)} (heap', lcl')\]

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

Rule for assignment to local:

\[(heap, lcl) \xrightarrow{e \triangleright v} (heap', lcl')\]

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

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

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

rule for multiplication (similar for other binary operators)

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

\[ (\text{heap, lcl} \cup \{x \mapsto 5\}) \xrightarrow{x \mapsto 5} (\text{heap, lcl} \cup \{x \mapsto 5\}) \]
\[ (\text{heap, lcl} \cup \{x \mapsto 5\}) \xrightarrow{\ast 2} (\text{heap, lcl} \cup \{x \mapsto 5\}) \]
\[ (\text{heap, lcl} \cup \{x \mapsto 5\}) \xrightarrow{\ast x \ast 2 \mapsto 10} (\text{heap, lcl} \cup \{x \mapsto 5\}) \]
\[ (\text{heap, lcl} \cup \{x \mapsto 5\}) \xrightarrow{x = x \ast 2 \mapsto 10} (\text{heap, lcl} \cup \{x \mapsto 10\}) \]
expression statement (assignment or method call):

\[
(\text{heap}, \text{lcl}) \xrightarrow{e \triangleright v} (\text{heap}', \text{lcl}')
\]

\[
(\text{heap}, \text{lcl}) \xrightarrow{e;} (\text{heap}', \text{lcl}')
\]

sequence of statements:

\[
(\text{heap}_1, \text{lcl}_1) \xrightarrow{s_1} (\text{heap}_2, \text{lcl}_2) \quad (\text{heap}_2, \text{lcl}_2) \xrightarrow{s_2} (\text{heap}_3, \text{lcl}_3)
\]

\[
(\text{heap}_1, \text{lcl}_1) \xrightarrow{s_1 \cdot s_2} (\text{heap}_3, \text{lcl}_3)
\]
Rules for Java Statements

if statement:

\[
\begin{align*}
(\text{heap}_1, \text{lcl}_1) & \xrightarrow{e \triangleright v} (\text{heap}_2, \text{lcl}_2) \quad (\text{heap}_2, \text{lcl}_2) & \xrightarrow{\text{bl}_1} (\text{heap}_3, \text{lcl}_3) \\
(\text{heap}_1, \text{lcl}_1) & \xrightarrow{\text{if}(e) \text{ bl}_1 \text{ else bl}_2} (\text{heap}_3, \text{lcl}_3)
\end{align*}
\]

, where \( v \neq 0 \)

\[
\begin{align*}
(\text{heap}_1, \text{lcl}_1) & \xrightarrow{e \triangleright v} (\text{heap}_2, \text{lcl}_2) \quad (\text{heap}_2, \text{lcl}_2) & \xrightarrow{\text{bl}_2} (\text{heap}_3, \text{lcl}_3) \\
(\text{heap}_1, \text{lcl}_1) & \xrightarrow{\text{if}(e) \text{ bl}_1 \text{ else bl}_2} (\text{heap}_3, \text{lcl}_3)
\end{align*}
\]

, where \( v = 0 \)