Verification Calculus Soundness

A fundamental problem!

informal language specification

proof rules

formal semantics
A fundamental problem!

- Informal language specification
- Proof rules
- Formal semantics
Verification Calculus Soundness

A fundamental problem!

informal language specification

proof rules  formal semantics
Validating Soundness of Proof Rules

**Bootstrapping**
Validate a core set of rules, generate and prove verification conditions for additional rules

**Cross-verification**
- against the BALI calculus for Java formalized in Isabelle/HOL
  [D. von Oheimb, T. Nipkow]
- against the Java semantics in the MAUDE system
  [J. Meseguer]

**Tests**
Using the compiler test suite Jacks
Validating Soundness of Proof Rules

**Bootstrapping**

Validate a core set of rules, generate and prove verification conditions for additional rules

**Cross-verification**

- against the BALI calculus for Java formalized in Isabelle/HOL
  
  [D. von Oheimb, T. Nipkow]

- against the Java semantics in the MAUDE system
  
  [J. Meseguer]

**Tests**

Using the compiler test suite Jacks
Validating Soundness of Proof Rules

**Bootstrapping**
Validate a core set of rules, generate and prove verification conditions for additional rules

**Cross-verification**
- against the BALI calculus for Java formalized in Isabelle/HOL
  [D. von Oheimb, T. Nipkow]
- against the Java semantics in the MAUDE system
  [J. Meseguer]

**Tests**
Using the compiler test suite Jacks
PostIncrementExpression:

PostfixExpression ++

At run time, if evaluation [...] completes abruptly, then the postfix increment expression completes abruptly and no incrementation occurs. Otherwise, the value 1 is added to the value of the variable and the sum is stored back into the variable. Before the addition, binary numeric promotion is performed on the value [...] The value of the postfix increment expression is the value of the variable before the new value is stored.
PostIncrementExpression:

PostfixExpression ++

At run time, if evaluation [...] completes abruptly, then the postfix increment expression completes abruptly and no incrementation occurs.

Otherwise, the value 1 is added to the value of the variable and the sum is stored back into the variable. Before the addition, binary numeric promotion is performed on the value [...] The value of the postfix increment expression is the value of the variable before the new value is stored.
Rule for Postfix Increment

**Intuitive rule (not correct!)**

\[ \implies \langle \pi \; x=y; \; y=y+1; \; \omega \rangle \phi \]

\[ \implies \langle \pi \; x=y++; \; \omega \rangle \phi \]

**But ...**

\[ x = 5 \implies \langle x=x++; \rangle (x = 6) \quad \text{INVALID} \]

**Correct rule**

\[ \implies \langle \pi \; v=y; \; y=y+1; \; x=v; \; \omega \rangle \phi \]

\[ \implies \langle \pi \; x=y++; \; \omega \rangle \phi \]
Rule for Postfix Increment

**Intuitive rule (not correct!)**

\[
\begin{align*}
\Longrightarrow & \quad \langle \pi \quad x=y; \quad y=y+1; \quad \omega \rangle \phi \\
\Longrightarrow & \quad \langle \pi \quad x=y++; \quad \omega \rangle \phi
\end{align*}
\]

**But ...**

\[x = 5 \Longrightarrow \langle x=x++; \rangle (x = 6) \quad \text{INVALID}\]

**Correct rule**

\[
\begin{align*}
\Longrightarrow & \quad \langle \pi \quad v=y; \quad y=y+1; \quad x=v; \quad \omega \rangle \phi \\
\Longrightarrow & \quad \langle \pi \quad x=y++; \quad \omega \rangle \phi
\end{align*}
\]
Rule for Postfix Increment

**Intuitive rule (not correct!)**

\[
\Rightarrow \langle \pi \ x=y; \ y=y+1; \ \omega \rangle \phi \\
\Rightarrow \langle \pi \ x=y++; \ \omega \rangle \phi
\]

**But ...**

\[
x = 5 \Rightarrow \langle x=x++; \rangle (x = 6) \quad \text{INVALID}
\]

**Correct rule**

\[
\Rightarrow \langle \pi \ v=y; \ y=y+1; \ x=v; \ \omega \rangle \phi \\
\Rightarrow \langle \pi \ x=y++; \ \omega \rangle \phi
\]
class T1241r1a {
    final int i=1; static final int j=1;
    static {
    }
}

class T1241r1b {
    /*@ public normal_behavior*/
    @ ensures \result == 7; @ */
    public static int main() {
        int s = 0; T1241r1a a = null;
        s = s + a.j;
        try {s = s + a.i;}
        catch (Exception e) {
            s = s + 2; a = new T1241r1a();
            s = s + a.i + 3;  }
        return s; }
}