Handling Java’s Abrupt Termination in a Sequent Calculus for Dynamic Logic

Bernhard Beckert
Bettina Sasse

UNIVERSITY OF KARLSRUHE
INSTITUTE FOR LOGIC, COMPLEXITY AND DEDUCTION SYSTEMS

i12www.ira.uka.de/~key

VerifiCard Workshop
Marseille, January 2002
Reasons for Limited Use of Verification

- No support for programming languages that are used in practice
Reasons for Limited Use of Verification

- No support for programming languages that are used in practice
- Verification requires knowledge in higher-order logic, tactic languages, etc.
- Verification is not integrated into standard CASE tools and software development processes
Reasons for Limited Use of Verification

- No support for programming languages that are used in practice
- Verification requires knowledge in higher-order logic, tactic languages, etc.
- Verification is not integrated into standard CASE tools and software development processes
Central Paradigm of the KeY Project

Formal methods must – and can – be integrated into commercially used methodologies, tools, and languages for software development.
Central Paradigm of the KeY Project

Formal methods must – and can – be integrated into commercially used methodologies, tools, and languages for software development.

Integrated tool for
- modelling
- formal specification
- verification

of object-oriented programs (Java Card)
The KeY System

CASE Tool

extension for formal specification

UML
OCL
Java

Verification Component

Dynamic Logic

Deduction Component

automated

interactive

counter examples
Dynamic Logic

Transparency of rules and proofs

- Formulas contain programs
- Basic rules for each programming construct
- Rule application corresponds to symbolic execution
Dynamic Logic

Transparency of rules and proofs
- Formulas contain programs
- Basic rules for each programming construct
- Rule application corresponds to symbolic execution

Handling “real” object-oriented language Java

Requires extensions and new concepts
Verification of Java Card: Difficulties

- Program state depends on the objects and their attributes
- Aliasing
- Polymorphism (dynamic binding)
- Evaluation of Java expressions may have side effects
- Programming constructs such as
  - abrupt termination (e.g. exceptions)
  - built-in data types (incl. arrays and strings)
  - initialisation of objects
Dynamic Logic

Syntax

- Modal operators $[p]$ and $\langle p \rangle$ for each program $p$
- Refer to the final state of $p$
Dynamic Logic

Syntax

- Modal operators $[p]$ and $\langle p \rangle$ for each program $p$
- Refer to the final state of $p$

Semantics

- $[p] F$: If $p$ terminates, then $F$ holds in the final state
  (partial correctness)
- $\langle p \rangle F$: $p$ terminates and $F$ holds in the final state
  (total correctness)
Expressivity of Dynamic Logic

Hoare triple

\[ F \rightarrow [p] G \quad \text{the same as} \quad \{F\} p \{G\} \]
Expressivity of Dynamic Logic

Hoare triple

\[ F \rightarrow [p] G \quad \text{the same as} \quad \{F\} \ p \ \{G\} \]

Simple example

\[ \forall n (\langle x = \text{is\_even}(n); \rangle x = \text{true} \rightarrow \langle x = \text{is\_even}(n+2); \rangle x = \text{true}) \]
Rule for if-else

premisses

\[ \Gamma, b = \text{true} \vdash \langle p \rangle F \quad \Gamma, b = \text{false} \vdash \langle q \rangle F \]

\[ \Gamma \vdash \langle \text{if} \,(b) \, \{p\} \, \text{else} \, \{q\} \rangle F \]

conclusion
Rule for if-else

**premisses**

\[ \Gamma, \ b = \text{true} \vdash \langle p \rangle F \]

\[ \Gamma, \ b = \text{false} \vdash \langle q \rangle F \]

\[ \Gamma \vdash \langle \text{if}(b) \ \{ p \} \ \text{else} \ \{ q \} \rangle F \]

**new proof obligation**

**conclusion**

**old proof obligation**
Abrupt Termination in Java

Reasons for abrupt termination

- **continue**  (with or w/o label)
  - loop  (current iteration)

- **break**  (with or w/o label)
  - loop, switch, labelled block

- **exception**
  - try-catch statement
    - (also: block, loop, method)

- **return**
  - method
    - (also: try-catch, block, loop)
Abrupt Termination in Java: Examples

Loop terminated by `break`

```java
while (true) {
    if (i==10) break;
    i++;
}
```
Abrupt Termination in Java: Examples

try-catch-finally with exception

```java
try {
    x = y / z;
} catch (ArithmeticException e) {
    x = 0;
} finally {
    z = z + 1;
}
```
New semantics for $\langle p \rangle F$:

$p$ terminates **normally (not abruptly)** and $F$ holds in the final state

There is no “return value” describing the reason for termination
possible contexts of an abrupt termination

- method
- block
- switch statement
- while, do-while, for loops
- try-catch-finally statement
Rule for while Loops

Symbolic execution of one loop iteration

\[ \Gamma \vdash \langle \text{if}(b)p \ \text{while}(b)\{p\}\rangle F \]

\[ \Gamma \vdash \langle \text{while}(b)\{p\}\rangle F \]
Rule for while Loops

Symbolic execution of one loop iteration

\[ \Gamma \vdash \langle \text{if}(b) l_1{:}{l_2{:}\{p'\}} \text{ while}(b){\{p\}} \rangle F \]

\[ \Gamma \vdash \langle \text{while}(b){\{p\}} \rangle F \]

Construction of \( p' \):

- `break` \( \rightarrow \) `break l_1`
- `continue` \( \rightarrow \) `break l_2`
Rule for while Loops: Example

```java
while (true) {
    if (i==10) break;
    i++;
}
```

VeriCard Workshop, Marseille, January 2002 – p.16
Rule for while Loops: Example

while (true) {
    if (i==10) break;
    i++;
}  \(p\)

\[\Gamma \vdash \langle \text{if(true)11:}\{\text{12:}\{\text{if(i==10) break 11; i++;}\}\} F \]

\[\Gamma \vdash \langle \text{while(true)\{if(i==10) break; i++;\}\} F \]
Rule for Exception that is Caught

\[ \Gamma \vdash \text{instanceof}(\text{exc}, T) \quad \Gamma \vdash \langle \text{try}\{e=\text{exc}; q\}\text{finally}\{r\}\rangle F \]

\[ \Gamma \vdash \langle \text{try}\{\text{throw} \\text{exc}; p\}\text{catch}(T \ e)\{q\}\text{finally}\{r\}\rangle F \]
try {throw exc; return 3;}
catch (Exception e) {return 4;}
finally {return 5;}

try {throw exc; return 3;}
catch (Exception e) {return 4;}
finally {return 5;}

Γ ⊢ instanceof (exc, Exception)

Γ ⊢ \{try{e=exc; return 4;}finally{return 5;}\} F

Γ ⊢ \{try{throw exc; return 3;} catch(Exception e){return 4;} finally{return 5;}\} F
Rule for Exception that is Caught: Example

\[ \Gamma \vdash \ldots \Gamma \vdash \langle \text{try}\{e=exc; \text{return} \ 4;\}\text{finally}\{\text{return} \ 5;\}\rangle F \]

\[ \Gamma \vdash \langle \text{try}\{\text{throw exc}; \text{return} \ 3;\}\text{catch}(\text{Exception} \ e)\{\text{return} \ 4;\}\text{finally}\{\text{return} \ 5;\}\rangle F \]
Rule for Exception that is Caught: Example

\[
\begin{align*}
\Gamma, e &= \text{exc} \vdash \langle \text{try} \{\text{return} 4;\} \text{finally} \{\text{return} 5;\} \rangle F \\
\Gamma &\vdash \ldots \\
\Gamma &\vdash \langle \text{try} \{\text{e=exc}; \text{return} 4;\} \text{finally} \{\text{return} 5;\} \rangle F \\
\Gamma &\vdash \langle \text{try} \{\text{throw} \text{exc}; \text{return} 3;\} \\
&\quad \text{catch(Exception e)}\{\text{return} 4;\} \\
&\quad \text{finally} \{\text{return} 5;\}\rangle F
\end{align*}
\]
Rule for Exception that is Caught: Example

\[
\Gamma, e = \text{exc} \vdash \langle \text{return } 5; \text{ return } 4; \rangle F
\]

\[
\Gamma, e = \text{exc} \vdash \langle \text{try}\{\text{return } 4;\}\text{finally}\{\text{return } 5;\} \rangle F
\]

\[
\Gamma \vdash \ldots
\]

\[
\Gamma \vdash \langle \text{try}\{\text{e=exc; return } 4;\}\text{finally}\{\text{return } 5;\} \rangle F
\]

\[
\Gamma \vdash \langle \text{try}\{\text{throw exc; return } 3;\}
\quad \text{catch(Exception e)}\{\text{return } 4;\}
\quad \text{finally}\{\text{return } 5;\} \rangle F
\]
Example

Proof obligation

while (true) {
    if (i==10) then break;
    i++;
}

terminates with $i = 10$ if started with $0 \leq i \leq 10$

Formal

$0 \leq i, i \leq 10 \vdash \langle p \rangle i = 10$
Example

\begin{align*}
\text{empty prog.} & \quad i = 10 \quad \text{induction hypothesis} \\
\text{break} & \\
\text{break} & \\
\text{if} & \\
\text{if} & \\
\text{while} & \\
\text{while} & \\
\text{if} & \\
\text{if} & \\
\text{if} & \\
\text{if} & \\
\text{while} & \\
\text{n} = 0 & \\
\text{n} \rightarrow n + 1 & \\
\therefore (\forall n)((n \leq 10 \land i = 10 - n) \rightarrow \langle p \rangle_i = 10) & \\
\text{choice of induction hypothesis} & \\
0 \leq i, \quad i \leq 10 & \therefore \langle p \rangle_i = 10
\end{align*}

\begin{verbatim}
while (true) {
  if (i==10) then
    break;
  i++;
}
\end{verbatim}