

Formal Methods for Java

Lecture 16: Abnormal Termination in Key

Jochen Hoenicke



Software Engineering
Albert-Ludwigs-University Freiburg

December 16, 2011

Abnormal Termination in Java

Abnormal termination in Java is caused by

- a `break` statement,
- a `continue` statement,
- a `return` statement,
- a `throw` statement, or
- a statement that throws an exception.

Abnormal Termination in Dynamic Logic

The formula $\langle \alpha \rangle \phi$ holds,

- iff α terminates **normally** and ϕ holds afterwards.

The formula $[\alpha] \phi$ holds,

- if α terminates **normally** and ϕ holds afterwards.
- if α terminates **abnormally**.
- if α does not terminate at all.

Reasoning about exceptions.

How can we express that statement α throws an exception?

- $\langle\{\alpha\}\rangle\phi$ is equivalent to **false** if α throws an exception or does not terminate
- $[\{\alpha\}]\phi$ is equivalent to **true** if α throws an exception or does not terminate
- The trick is to put an exception handler into the code:

```
 $\langle\{\text{Throwable } thrown = null;$   
  try  $\{\alpha; \}$   
  catch  $(\text{Throwable } ex)\{thrown = ex; \}\}\rangle thrown \neq null$ 
```

Reasoning with try-catch blocks

Many DL-rules in KeY just skip opening of try blocks, e.g.

$\backslash find(\backslash \langle \{ \dots \#loc = \#se \dots \} \rangle \backslash \rangle post)$

$\backslash replacewith(\{ \#loc := \#se \} \backslash \langle \{ \dots \} \rangle \backslash \rangle post)$

Here \dots stands for an arbitrary number of opening try-blocks, labelled blocks and normal blocks.

Example:

$\langle \langle \mathbf{try} \{ label : \{ \mathbf{try} \{ x = 5 \dots \} \} \} \rangle \rangle \phi$

is replaced with

$\{ x := 5 \} \langle \langle \mathbf{try} \{ label : \{ \mathbf{try} \{ \dots \} \} \} \rangle \rangle \phi$

Reasoning with try-catch blocks (2)

When an exception is thrown, the surrounding try blocks become important:

```
\find( \< { .. try { throw #se; #slist1 }  
        catch (#t #v0) { #slist2 } ... } \> post )
```

- 1 throwing a handled exception: #se instanceof #t

```
\replacewith( \< { .. #t #v0 = #se; #slist2 ... } \> post )
```
- 2 throwing an unhandled exception: ! (#se instanceof #t)

```
\replacewith( \< { .. throw #se; ... } \> post )
```
- 3 throwing a null pointer: #se = null

```
\replacewith( \< { .. try { throw new NullPointerException(); #slist1  
        catch (#t #v0) { #slist2 } ... } \> post )
```

The KeY system defines a single rule:

```
\replacewith( \< { .. if (#se = null) then  
        try { throw new NullPointerException(); #slist1  
        catch (#t #v0) { #slist2 }  
        else if (#se instanceof #t) then  
            #t v0 = #se; #slist2  
        else throw #se;  
        ... } \> post )
```