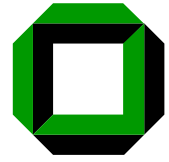
Handling Java's Abrupt Termination in a Sequent Calculus for Dynamic Logic

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

Verifier and software developer speak different languages

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

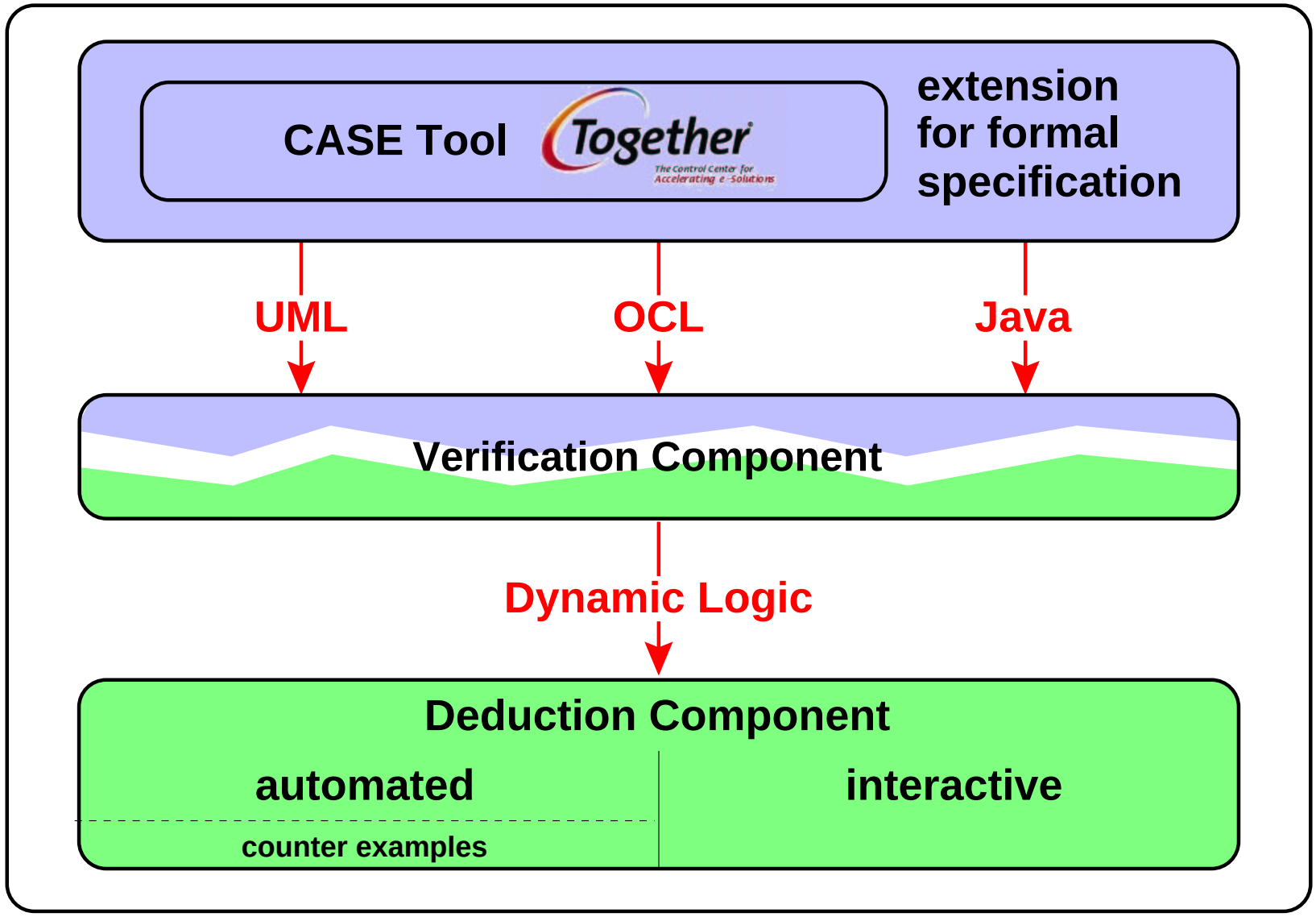
Formal methods must – and can – be integrated into commercially used methodologies, tools, and languages for software development

Integrated tool for

- **modelling**
- **formal specification**
- **verification**

of object-oriented programs (Java Card)

The KeY System



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

$F \longrightarrow [p]\, G$ **the same as** $\{F\}\ p\ \{G\}$

Expressivity of Dynamic Logic

Hoare triple

$F \rightarrow [p]\, G$ **the same as** $\{F\}\ p\ \{G\}$

Simple example

$$\forall n\,(\langle \texttt{x = is\_even(}n\texttt{);}\rangle \texttt{x} = \texttt{true} \rightarrow \langle \texttt{x = is\_even(}n+2\texttt{);}\rangle \texttt{x} = \texttt{true})$$

Rule for if-else

premisses

$$\frac{\Gamma,\ b = \texttt{true} \vdash \langle p \rangle F \qquad \Gamma,\ b = \texttt{false} \vdash \langle q \rangle F}{\Gamma \vdash \langle \texttt{if(}b\texttt{) \{}p\texttt{\} else \{}q\texttt{\}} \rangle F}$$

conclusion

Rule for if-else

premisses

new proof obligation

$$\frac{\Gamma,\ b = \texttt{true} \vdash \langle p \rangle F \qquad \Gamma,\ b = \texttt{false} \vdash \langle q \rangle F}{\Gamma \vdash \langle \texttt{if(}b\texttt{) \{}p\texttt{\} else \{}q\texttt{\}} \rangle F}$$

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`

```
while (true) {

    if (i==10) break;

    i++;

}
```

Abrupt Termination in Java: Examples

`try-catch-finally` with exception

```
try {
    x=y/z;
} catch(ArithmeticException e) {
    x = 0;
} finally {
    z = z+1;
}
```

Integrating Abrupt Termination into DL

- **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**

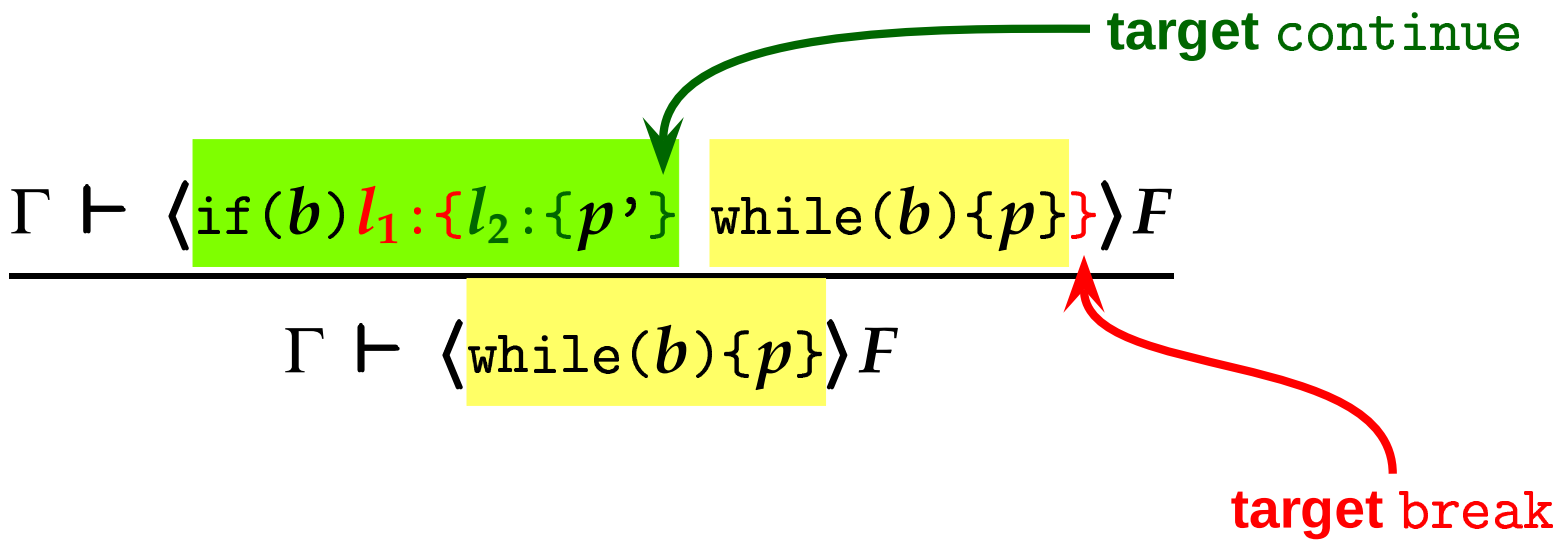
Rule for while Loops

Symbolic execution of one loop iteration

$$\frac{\Gamma \vdash \langle \texttt{if}(b)\, p \;\; \texttt{while}(b)\{p\}\rangle F}{\Gamma \vdash \langle \texttt{while}(b)\{p\}\rangle F}$$

Rule for while Loops

Symbolic execution of one loop iteration



Construction of p':

`break`	$\longrightarrow$	`break` l_1
`continue`	$\longrightarrow$	`break` l_2

Rule for while Loops: Example

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

p

Rule for while Loops: Example

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

p

$$\frac{\Gamma \vdash \langle \texttt{if(true)l1:\{l2:\{if(i==10) break l1; i++;\}while(true)\{}p\texttt{\}\}} \rangle F}{\Gamma \vdash \langle \texttt{while(true)\{if(i==10) break; i++;\}} \rangle F}$$

Rule for Exception that is Caught

$$\frac{\Gamma \vdash \mathit{instanceof}(\mathit{exc}, T) \qquad \Gamma \vdash \langle \texttt{try\{}e\texttt{=}\mathit{exc}\texttt{;}\ q\texttt{\}finally\{}r\texttt{\}}\rangle F}{\Gamma \vdash \langle \texttt{try\{throw }\mathit{exc}\texttt{;}\ p\texttt{\}catch(}T\ e\texttt{)\{}q\texttt{\}finally\{}r\texttt{\}}\rangle F}$$

Rule for Exception that is Caught: Example

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

Rule for Exception that is Caught: Example

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

$$\frac{\Gamma \vdash \mathit{instanceof}(\texttt{exc}, \texttt{Exception}) \qquad \Gamma \vdash \langle \texttt{try\{e=exc; return 4;\}finally\{return 5;\}} \rangle F}{\Gamma \vdash \langle \texttt{try\{throw exc; return 3;\} catch(Exception e)\{return 4;\} finally\{return 5;\}} \rangle F}$$

Rule for Exception that is Caught: Example

$$\frac{\Gamma \vdash \ldots \qquad \Gamma \vdash \langle \texttt{try\{e=exc; return 4;\}finally\{return 5;\}} \rangle F}{\Gamma \vdash \langle \texttt{try\{throw exc; return 3;\}} \; \texttt{catch(Exception e)\{return 4;\}} \; \texttt{finally\{return 5;\}} \rangle F}$$

Rule for Exception that is Caught: Example

$$\dfrac{\dfrac{\Gamma, \texttt{e} = \texttt{exc} \vdash \langle \texttt{try\{return 4;\}finally\{return 5;\}} \rangle F}{\Gamma \vdash \langle \texttt{try\{e=exc; return 4;\}finally\{return 5;\}} \rangle F} \quad \Gamma \vdash \ldots}{\Gamma \vdash \langle \texttt{try\{throw exc; return 3;\}} \; \texttt{catch(Exception e)\{return 4;\}} \; \texttt{finally\{return 5;\}} \rangle F}$$

Rule for Exception that is Caught: Example

$$\dfrac{\dfrac{\dfrac{\Gamma, \texttt{e} = \texttt{exc} \vdash \langle \texttt{return 5; return 4;} \rangle F}{\Gamma, \texttt{e} = \texttt{exc} \vdash \langle \texttt{try\{return 4;\}finally\{return 5;\}} \rangle F}}{\Gamma \vdash \langle \texttt{try\{e=exc; return 4;\}finally\{return 5;\}} \rangle F} \quad \Gamma \vdash \ldots}{\Gamma \vdash \langle \texttt{try\{throw exc; return 3;\}} \; \texttt{catch(Exception e)\{return 4;\}} \; \texttt{finally\{return 5;\}} \rangle F}$$

Example

Proof obligation

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

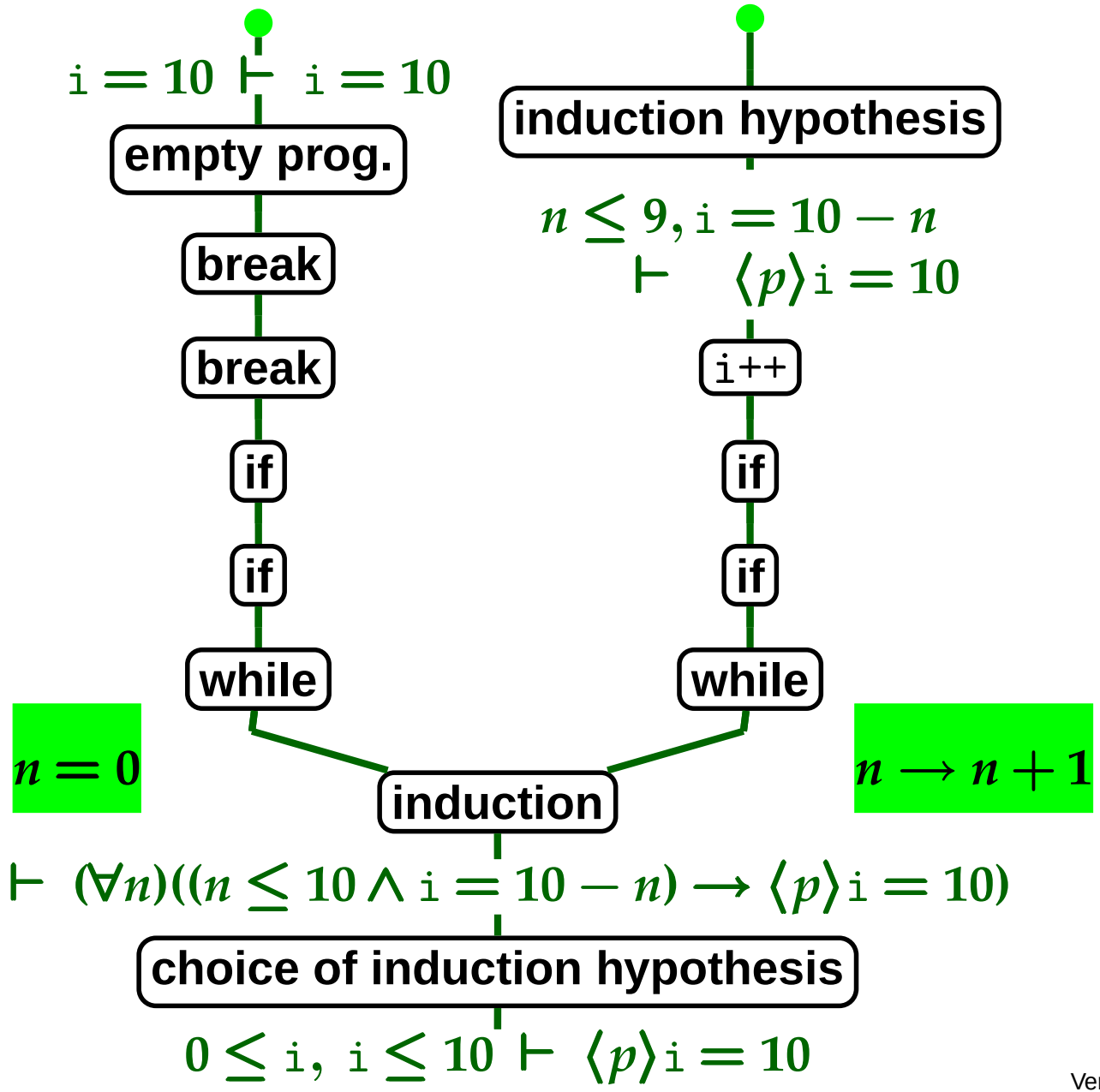
$\}\ p$

terminates with $\mathtt{i} = 10$ if started with $0 \leq \mathtt{i} \leq 10$

Formal

$$0 \leq \mathtt{i},\ \mathtt{i} \leq 10 \vdash \langle p \rangle \mathtt{i} = 10$$

Example




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