Structure and Semantic Properties of Extracted Specifications

Bernhard Beckert, Christoph Gladisch

www.key-project.org

Deduktionstreffen 2007

Koblenz, Germany
March 26, 2007
The KeY System

Overview
- Automatic and Interactive Prover
- 100% JavaCard

First-Order Dynamic Logic

\[ \langle p \rangle \phi \]

(Implication) \( \psi \rightarrow [p]\phi \)

\{\psi\} p \{\phi\} (Hoare triple)
The KeY System

Overview
- Automatic and Interactive Prover
- 100% JavaCard

First-Order Dynamic Logic

\( [p] \phi \)
\( \langle p \rangle \phi \)
(Implication) \( \psi \rightarrow [p] \phi \)
\{ \psi \} p \{ \phi \} \) (Hoare triple)
The KeY System

### Overview
- Automatic and Interactive Prover
- 100% JavaCard

### First-Order Dynamic Logic

- \([p]\phi\)
- \(\langle p\rangle\phi\)

(Implication) \(\psi \rightarrow [p]\phi\)

\(\{\psi\}p\{\phi\}\) (Hoare triple)
The KeY System

**Overview**
- Automatic and Interactive Prover
- 100% JavaCard

**First-Order Dynamic Logic**

\[
[p] \phi \\
\langle p \rangle \phi
\]

(Implication) \( \psi \rightarrow [p] \phi \) \hspace{1cm} \{ \psi \} p \{ \phi \} \ (Hoare \ triple)
The KeY System

**Overview**
- Automatic and Interactive Prover
- 100% JavaCard

**First-Order Dynamic Logic**

<table>
<thead>
<tr>
<th>$[p] \phi$</th>
<th>$\langle p \rangle \phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Implication) $\psi \rightarrow [p] \phi$</td>
<td>${ \psi } p { \phi }$ (Hoare triple)</td>
</tr>
</tbody>
</table>
Roles of Specifications

**Requirement Specification**
- Given by the user
- To be tested or verified against the target program

**Full Specification**
- Must comply with the target program
- Automatically or interactively extracted
Roles of Specifications

**Requirement Specification**
- Given by the user
- To be tested or verified against the target program

**Full Specification**
- Must comply with the target program
- Automatically or interactively extracted
Roles of Specifications

Requirement Specification
- Given by the user
- To be tested or verified against the target program

Full Specification
- Must comply with the target program
- Automatically or interactively extracted
Roles of Specifications

### Requirement Specification
- Given by the user
- To be tested or verified against the target program

### Full Specification
- Must comply with the target program
- Automatically or interactively extracted
Roles of Specifications

<table>
<thead>
<tr>
<th>Requirement Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Given by the user</td>
</tr>
<tr>
<td>- To be tested or verified against the target program</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Full Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Must comply with the target program</td>
</tr>
<tr>
<td>- Automatically or interactively extracted</td>
</tr>
</tbody>
</table>
Roles of Specifications

**Requirement Specification**
- Given by the user
- To be tested or verified against the target program

**Full Specification**
- Must comply with the target program
- Automatically or interactively extracted
Application 1 (Verification)

(Extracted) Full Specification

(KeY) Specification Extraction

Source code
Application 1 (Verification)

1. Requirement Specification
2. (Extracted) Full Specification
3. (KeY) Specification Extraction
4. Source code
Application 1 (Verification)

Requirement Specification → (Extracted) Full Specification → FOL Prover

(KeY) Specification Extraction → Source code
Application 1 (Verification)

Application 2 (Testing)

Requirement Specification → (Extracted) Full Specification → (KeY) Specification Extraction → Source code
Application 2 (Testing)

Requirement Specification → Black-box Testing Tool

(Extracted) Full Specification

(KeY) Specification Extraction

Source code

{ White-box Testing }
Application 3 (Contracts, Program Replacements)

```
Input_p → A  Sub  B : Output_p
```

P
Application 3 (Contracts, Program Replacements)
Application 3 (Contracts, Program Replacements)

\[
\begin{align*}
\text{Input}_{\text{Sub}} & \rightarrow \text{Sub} \\
\text{Output}_{\text{Sub}} & : \\
\text{Input}_p & \rightarrow \text{A} \\
\text{B} & : \text{Output}_p
\end{align*}
\]
Application 3 (Contracts, Program Replacements)

\[ \text{Input}_P \rightarrow A : \text{Input}_{\text{Sub}} \quad \text{Output}_{\text{Sub}} \rightarrow B : \text{Output}_P \]
Application 3 (Contracts, Program Replacements)
Application 3 (Contracts, Program Replacements)
Application 3 (Contracts, Program Replacements)

```
<table>
<thead>
<tr>
<th>Pre</th>
<th>Method</th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ind(n)</td>
<td>Loop</td>
<td>Ind(0)</td>
</tr>
</tbody>
</table>

Input_p → Input_{Sub} → Output_{Sub} → B → Output_p
```

P
Application 3 (Contracts, Program Replacements)
Semantic Properties (Examples)

Many details have to be recognized and handled with care.

**Pre and Post Condition vs. Pre and Post State**

\[ \text{pre} \rightarrow [\alpha]\text{post} \equiv [\alpha](\text{pre} \rightarrow \text{post}) \]

**Termination: Box vs. Diamond**

\[ \text{false} \rightarrow \langle nt. \rangle \text{post} \neq \langle nt. \rangle (\text{false} \rightarrow \text{post}) \]

**Semantics of Formulas but with different Signatures**

<table>
<thead>
<tr>
<th>DL:</th>
<th>FOL:</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{pre} \rightarrow \langle \alpha \rangle \text{post}</td>
<td>\text{pre}' \rightarrow \text{post}'</td>
</tr>
<tr>
<td>$\phi_{DL}$ $\Leftrightarrow$ $\phi_{FOL}$</td>
<td>?</td>
</tr>
</tbody>
</table>
Many details have to be recognized and handled with care.

**Pre and Post Condition vs. Pre and Post State**

\[ \text{pre} \rightarrow [\alpha]\text{post} \equiv [\alpha](\text{pre} \rightarrow \text{post}) \]

**Termination: Box vs. Diamond**

\[ \text{false} \rightarrow \langle nt. \rangle \text{post} \neq \langle nt. \rangle (\text{false} \rightarrow \text{post}) \]

**Semantics of Formulas but with different Signatures**

<table>
<thead>
<tr>
<th>DL</th>
<th>FOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>pre \rightarrow \langle \alpha \rangle post</td>
<td>pre' \rightarrow post'</td>
</tr>
</tbody>
</table>

\[ \phi_{DL} \iff \phi_{FOL} \]
Many details have to be recognized and handled with care.

### Pre and Post Condition vs. Pre and Post State

\[ pre \rightarrow [\alpha]post \equiv [\alpha](pre \rightarrow post) \]

### Termination: Box vs. Diamond

\[ false \rightarrow \langle nt. \rangle post \not\equiv \langle nt. \rangle(false \rightarrow post) \]

### Semantics of Formulas but with different Signatures

**DL:**

\[ pre \rightarrow \langle \alpha \rangle post \]

**FOL:**

\[ pre' \rightarrow post' \]

\[ \phi_{DL} \iff \phi_{FOL} \]
Many details have to be recognized and handled with care.

**Pre and Post Condition vs. Pre and Post State**

\[
pre \to [\alpha] post \equiv [\alpha] (pre \to post)
\]

**Termination: Box vs. Diamond**

\[
false \to \langle nt. \rangle post \not\equiv \langle nt. \rangle (false \to post)
\]

**Semantics of Formulas but with different Signatures**

<table>
<thead>
<tr>
<th>DL:</th>
<th>FOL:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pre) \to \langle\alpha\rangle post</td>
<td></td>
</tr>
<tr>
<td>(pre') \to (post')</td>
<td></td>
</tr>
<tr>
<td>(\phi_{DL}) \iff (\phi_{FOL})</td>
<td></td>
</tr>
</tbody>
</table>

?
Many details have to be recognized and handled with care.

**Pre and Post Condition vs. Pre and Post State**

\[
pre \rightarrow [\alpha]post \equiv [\alpha](pre \rightarrow post)
\]

**Termination: Box vs. Diamond**

\[
false \rightarrow \langle nt. \rangle post \not\equiv \langle nt. \rangle(false \rightarrow post)
\]

**Semantics of Formulas but with different Signatures**

<table>
<thead>
<tr>
<th>DL:</th>
<th>(pre)</th>
<th>(\langle \alpha \rangle post)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOL:</td>
<td>(pre')</td>
<td>(post')</td>
</tr>
</tbody>
</table>

\(\phi_{DL} \iff \phi_{FOL}\)
Many details have to be recognized and handled with care.

### Pre and Post Condition vs. Pre and Post State

\[ pre \rightarrow [\alpha]post \equiv [\alpha](pre \rightarrow post) \]

### Termination: Box vs. Diamond

\[ false \rightarrow \langle nt.\rangle post \not\equiv \langle nt.\rangle(false \rightarrow post) \]

### Semantics of Formulas but with different Signatures

- **DL:** 
  \[ pre \rightarrow \langle \alpha \rangle post \]

- **FOL:** 
  \[ pre' \rightarrow post' \]

\[ \phi_{DL} \iff \phi_{FOL} \]
Semantic Properties (Examples)

Many details have to be recognized and handled with care.

**Pre and Post Condition vs. Pre and Post State**

\[ pre \rightarrow [\alpha] post \equiv [\alpha](pre \rightarrow post) \]

**Termination: Box vs. Diamond**

\[ false \rightarrow \langle nt. \rangle post \neq \langle nt. \rangle (false \rightarrow post) \]

**Semantics of Formulas but with different Signatures**

- DL: \[ pre \rightarrow \langle \alpha \rangle post \]
- FOL: \[ pre' \rightarrow post' \]

\[ \phi_{DL} \iff \phi_{FOL} \]
Many details have to be recognized and handled with care.

**Pre and Post Condition vs. Pre and Post State**

\[ pre \rightarrow [\alpha]post \equiv [\alpha](pre \rightarrow post) \]

**Termination: Box vs. Diamond**

\[ false \rightarrow \langle nt. \rangle post \not\equiv \langle nt. \rangle(false \rightarrow post) \]

**Semantics of Formulas but with different Signatures**

**DL:**

\[ pre \rightarrow \langle \alpha \rangle post \]

**FOL:**

\[ pre' \rightarrow post' \]

\[ \phi_{DL} \iff \phi_{FOL} \]
Specification Extraction (Syntactic properties)

Proof Tree

\[
\begin{align*}
\frac{(B_1)}{\Gamma', x - y \geq 0 \Rightarrow} & \quad \frac{(B_2)}{\{d := x - y\}[\text{if} \ldots] \Phi} \\
\frac{(B_3)}{\Gamma', x - y < 0 \Rightarrow \ldots} & \quad \frac{(B_4)}{\ldots} \\
\Gamma \Rightarrow [\text{if}(x-y<0)d=x-y;\text{else}d=y-x\ldots]d_{\text{post}} = d \ldots
\end{align*}
\]
Specification Extraction (Syntactic properties)

**Proof Tree**

\[
\begin{align*}
&(B_1) \quad (B_2) \\
\Gamma' \quad x - y \geq 0 &\Rightarrow \\
\{d := x - y\}[\text{if \ldots}]\Phi \\
&(B_3) \quad (B_4) \\
\Gamma' \quad x - y < 0 &\Rightarrow \ldots \\
\ldots &\\
\Gamma &\Rightarrow [\text{if}(x-y<0)d=x-y;\text{else}d=y-x\ldots]d_{\text{post}} = d\ldots
\end{align*}
\]

**Specification:**

- \(B_1 \land B_2 \land B_3 \land B_4\)
- Contains program structure
**Specification Extraction (Syntactic properties)**

**Proof Tree**

\[
\frac{(B_1) \quad (B_2)}{(B_3) \quad (B_4)}
\]

\[
\Gamma', x - y \geq 0 \Rightarrow \{d := x - y\}[\text{if } \ldots ]\Phi
\]

\[
\Gamma' \Rightarrow [\text{if}(x-y<0)d=x-y;\text{else}d=y-x\ldots]d_{\text{post}} = d\ldots
\]

**Specification:**

- \(B_1 \land B_2 \land B_3 \land B_4\)
- Contains program structure
Every DL-formula can be translated into a FOL-formula with Integers.

Example: the DL-formula $y = z \rightarrow \{x = y\}z = x$ can be translated into:

- true
- $\neg y = z \lor z = y$
- $\forall : 2, \neg : 1, = : 3, x : 4, y : 5, z : 6$
  
  \[2 \times 10^0 + 1 \times 10^1 + 3 \times 10^2 + 5 \times 10^3 + 6 \times 10^4 + 3 \times 10^5 + 6 \times 10^6 + 5 \times 10^7\]

IsTaut(21356365), sub(0, 21356365) = 1356, sub(1, 21356365) = 365, op(x) = xmod10, op(21356365) = 2

(op(x) = 2 $\rightarrow$ IsTaut(sub(0, x)) $\lor$ IsTaut(sub(1, x)) $\rightarrow$ IsTaut(x)
Structure of Specifications

**Specification Extraction**

Every DL-formula can be translated into a FOL-formula with Integers.

Example: the DL-formula $y = z \rightarrow \{x = y\}z = x$ can be translated into:

- $\text{true}$
- $\neg y = z \vee z = y$
- $\forall : 2, \neg : 1, = : 3, x : 4, y : 5, z : 6$
  
  \[
  2 \times 10^0 + 1 \times 10^1 + 3 \times 10^2 + 5 \times 10^3 + 6 \times 10^4 + 3 \times 10^5 + 6 \times 10^6 + 5 \times 10^7
  \]

  $\text{IsTaut}(21356365), \text{sub}(0, 21356365) = 1356, \text{sub}(1, 21356365) = 365, \text{op}(x) = x \text{mod} 10, \text{op}(21356365) = 2$

  $(\text{op}(x) = 2 \rightarrow \text{IsTaut}(\text{sub}(0, x)) \vee \text{IsTaut}(\text{sub}(1, x)) \rightarrow \text{IsTaut}(x)$
Every DL-formula can be translated into a FOL-formula with Integers.

Example: the DL-formula $y = z \rightarrow \{x = y\}z = x$ can be translated into:

- true
- $\neg y = z \lor z = y$

$\forall : 2, \neg : 1, =: 3, x : 4, y : 5, z : 6$

$2 \cdot 10^0 + 1 \cdot 10^1 + 3 \cdot 10^2 + 5 \cdot 10^3 + 6 \cdot 10^4 + 3 \cdot 10^5 + 6 \cdot 10^6 + 5 \cdot 10^7$\n
$\text{IsTaut}(21356365), \text{sub}(0, 21356365) = 1356, \text{sub}(1, 21356365) = 365, \text{op}(x) = x \text{mod} 10, \text{op}(21356365) = 2$\n
$(\text{op}(x) = 2 \rightarrow \text{IsTaut}(\text{sub}(0, x)) \lor \text{IsTaut}(\text{sub}(1, x)) \rightarrow \text{IsTaut}(x)$
Structure of Specifications

**Specification Extraction**

Every DL-formula can be translated into a FOL-formula with Integers.

Example: the DL-formula $y = z \rightarrow \{x = y\}z = x$ can be translated into:

- true
- $\neg y = z \lor z = y$
- $\forall : 2, \neg : 1, = : 3, x : 4, y : 5, z : 6$
  
  $2 \times 10^0 + 1 \times 10^1 + 3 \times 10^2 + 5 \times 10^3 + 6 \times 10^4 + 3 \times 10^5 + 6 \times 10^6 + 5 \times 10^7$

  $\text{IsTaut}(21356365), \text{sub}(0, 21356365) = 1356, \ \text{sub}(1, 21356365) = 365, \ \text{op}(x) =$
  
  $x \mod 10, \ \text{op}(21356365) = 2$

  $(\text{op}(x) = 2 \rightarrow \text{IsTaut}(\text{sub}(0, x)) \lor \text{IsTaut}(\text{sub}(1, x))) \rightarrow \text{IsTaut}(x)$
Conclusion

- Many details involved
- The quality of a Tool Chain depends on the structure of a specification
- Interesting question: how to extract good specifications
- The KeY System can extract specifications
Conclusion

- Many details involved
  - The quality of a Tool Chain depends on the structure of a specification
  - Interesting question: how to extract good specifications
  - The KeY System can extract specifications
Conclusion

- Many details involved
- The quality of a Tool Chain depends on the structure of a specification
- Interesting question: how to extract good specifications
- The KeY System can extract specifications
Conclusion

- Many details involved
- The quality of a Tool Chain depends on the structure of a specification
- Interesting question: how to extract good specifications
- The KeY System can extract specifications
Conclusion

- Many details involved
- The quality of a Tool Chain depends on the structure of a specification
- Interesting question: how to extract good specifications
- The KeY System can extract specifications