Generation of Proof Obligations to Ensure the Soundness of Taclets

Philipp Rümmer
ph.r@gmx.net

University of Karlsruhe
Institute for Logic, Complexity and Deduction Systems
D-76128 Karlsruhe, Germany
Overview

- Introduction to taclets
Overview

- Introduction to taclets
- Concept to ensure the soundness of taclets
Overview

- Introduction to taclets
- Concept to ensure the soundness of taclets
- Examples
Introduction to Taclets

- Method to define rules of sequent calculi
Introduction to Taclets

- Method to define rules of sequent calculi
- Introduced by Elmar Habermalz in: *Ein dynamisches automatisierbares interaktives Kalkül für schematische theoriespezifische Regeln* (PhD thesis)
Introduction to Taclets

- Method to define rules of sequent calculi
- Introduced by Elmar Habermalz in: *Ein dynamisches automatisierbares interaktives Kalkül für schematische theoriespezifische Regeln* (PhD thesis)
- In KeY: Majority of rules defined through taclets
• Taclet in concrete syntax:

\[
\begin{align*}
    & \text{if } (\text{ifseq}) \text{ find } (f) \\
    & \quad \text{replacewith } (rw_1) \text{ add } (add_1); \\
    & \quad \vdots \\
    & \quad \text{replacewith } (rw_k) \text{ add } (add_k)
\end{align*}
\]
• Taclet in concrete syntax:
  \[
  \text{if } (\text{ifseq}) \text{ find } (f) \\
  \quad \text{replacewith } (rw_1) \text{ add } (add_1); \\
  \vdots \\
  \quad \text{replacewith } (rw_k) \text{ add } (add_k)
  \]

• For instance: Modus ponens
  \[
  \text{if } (\phi \vdash) \text{ find } (\vdash \phi \rightarrow \psi) \\
  \quad \text{replacewith } (\vdash \psi)
  \]
For first-order logic:
For first-order logic:
  - Terms
Schema Variables

- For first-order logic:
  - Terms
  - Formulas
**Schema Variables**

- For first-order logic:
  - Terms
  - Formulas
  - Object variables

- For JavaCardDL in addition:
  - Program variables
  - Java statements
  - Java expressions
Schema Variables

- For first-order logic:
  - Terms
  - Formulas
  - Object variables
- For JavaCardDL in addition:
Schema Variables

- For first-order logic:
  - Terms
  - Formulas
  - Object variables
- For JavaCardDL in addition:
  - Program variables
Schema Variables

- For first-order logic:
  - Terms
  - Formulas
  - Object variables
- For JavaCardDL in addition:
  - Program variables
  - Java statements
Schema Variables

- For first-order logic:
  - Terms
  - Formulas
  - Object variables
- For JavaCardDL in addition:
  - Program variables
  - Java statements
  - Java expressions
Soundness of Calculi

- Sequent $\Gamma \vdash \Delta$ is called valid iff

\[ \bigwedge \Gamma \rightarrow \bigvee \Delta \]

is valid
Soundness of Calculi

- Sequent $\Gamma \vdash \Delta$ is called valid iff
  $\bigwedge \Gamma \rightarrow \bigvee \Delta$
  is valid
- A calculus is sound iff only valid sequents can be derived
Soundness of Rules

- Sufficient criterion for soundness of calculus: Rule applications preserve validity
Soundness of Rules

- Sufficient criterion for soundness of calculus: Rule applications preserve validity

\[
P_1 \quad P_2 \quad \cdots \quad P_k \\
\hline
Q
\]

\[P_1, P_2, \ldots, P_k \text{ valid } \implies Q \text{ valid}\]
Concept to Ensure Soundness of Taclets

Method introduced by Elmar Habermalz:
Concept to Ensure Soundness of Taclets

Method introduced by Elmar Habermalz:

Taclet $t$
Concept to Ensure Soundness of Taclets

Method introduced by Elmar Habermalz:

Taclet $t$

$\downarrow$

Meaning formula $M(t)$
Concept to Ensure Soundness of Taclets

Method introduced by Elmar Habermalz:

Taclet $t$

\downarrow

Meaning formula $M(t)$

\downarrow

Proof obligation $M_{Sk}(t)$
Concept to Ensure Soundness of Taclets

Method introduced by Elmar Habermalz:

Taclet \( t \)

\[ \downarrow \]

Meaning formula \( M(t) \)

\[ \downarrow \]

Proof obligation \( M_{Sk}(t) \)

\[ M_{Sk}(t) \text{ valid} \iff t \text{ sound} \]
Example of Meaning Formula

- Taclet $t_1$ exchanging quantifiers:

  \[
  \text{find ( } \vdash \forall y. \forall x. \phi) \\
  \text{replacewith ( } \vdash \forall x. \forall y. \phi)
  \]
Example of Meaning Formula

- Taclet $t_1$ exchanging quantifiers:
  
  \[
  \text{find } ( \vdash \forall y. \forall x. \phi) \\
  \text{replacewith } ( \vdash \forall x. \forall y. \phi)
  \]

- $M(t_1) = \forall x. \forall y. \phi \rightarrow \forall y. \forall x. \phi$
Example of Meaning Formula (2)

- Taclet $t_2$ splitting an if-statement:

\[
\text{find}(\langle l : \text{if } (x==0) \#s \text{ else } \#t \rangle \phi) \\
\text{replacewith}(\langle l : \#s \rangle \phi) \text{ add}(x \div 0 \mid - ) ; \\
\text{replacewith}(\langle l : \#t \rangle \phi) \text{ add}( \mid x \div 0 )
\]
Example of Meaning Formula (2)

\[
\text{find}(\langle l : \text{if } (x==0) \#s \text{ else } \#t \rangle \phi)
\]

\[
\text{replacewith}(\langle l : \#s \rangle \phi) \quad \text{add}(x \div 0 \vdash );
\]

\[
\text{replacewith}(\langle l : \#t \rangle \phi) \quad \text{add}(\vdash x \div 0)
\]
Example of Meaning Formula (2)

\[
\begin{align*}
\text{find}(\langle l: \text{if } (x==0) \ #s \ \text{else } \ #t \rangle \phi) \\
\quad \text{replacewith}(\langle l: \ #s \rangle \phi) \ \text{add}(x \div 0 \ |- ) \\
\quad \text{replacewith}(\langle l: \ #t \rangle \phi) \ \text{add}(\ |- \ x \div 0)
\end{align*}
\]

\[M(t_2) = \]

\[(x \div 0 \land \]
\[(\langle l: \text{if } (x==0) \ #s \ \text{else } \ #t \rangle \phi \leftrightarrow \langle l: \ #s \rangle \phi) \]

\[\lor (\neg (x \div 0) \land \]
\[(\langle l: \text{if } (x==0) \ #s \ \text{else } \ #t \rangle \phi \leftrightarrow \langle l: \ #t \rangle \phi) \]
Taclet Proof Obligations

- Schema variables of meaning formulas are replaced with skolem symbols
• Schema variables of meaning formulas are replaced with skolem symbols

SVs for terms $\rightarrow$ function symbols
Taclet Proof Obligations

- Schema variables of meaning formulas are replaced with skolem symbols

  SVs for terms → function symbols
  SVs for formulas → predicate symbols
Taclet Proof Obligations

- Schema variables of meaning formulas are replaced with skolem symbols

  SVs for terms → function symbols
  SVs for formulas → predicate symbols
  SVs for logical variables → logical variables
Schema variables of meaning formulas are replaced with skolem symbols

- SVs for terms → function symbols
- SVs for formulas → predicate symbols
- SVs for logical variables → logical variables
- SVs for program var. → program variables
Taclet Proof Obligations

- Schema variables of meaning formulas are replaced with skolem symbols

  SVs for terms → function symbols
  SVs for formulas → predicate symbols
  SVs for logical variables → logical variables
  SVs for program var. → program variables
  SVs for statements → atomic programs
Taclet Proof Obligations

- Schema variables of meaning formulas are replaced with skolem symbols

  SVs for terms → function symbols
  SVs for formulas → predicate symbols
  SVs for logical variables → logical variables
  SVs for program var. → program variables
  SVs for statements → atomic programs
  SVs for expressions → “atomic expr.”
Example of Proof Obligation

- Meaning formula of $t_1$ is

\[
M(t_1) = \forall x. \forall y. \phi \rightarrow \forall y. \forall x. \phi
\]
Example of Proof Obligation

- Meaning formula of $t_1$ is
  
  $$M(t_1) = \forall x.\forall y. \phi \rightarrow \forall y.\forall x. \phi$$

- Taclet proof obligation:
  
  $$M_{Sk}(t_1) = \forall u.\forall v. p_{Sk}(u, v) \rightarrow \forall v.\forall u. p_{Sk}(u, v)$$
Example of Proof Obligation (2)

- Meaning formula of \( t_2 \) is

\[
(x \div 0 \land
\langle l : \text{if } (x==0) \#s \text{ else } \#t \phi \leftrightarrow \langle l : \#s \phi \rangle \rangle
\lor
\neg (x \div 0) \land
\langle l : \text{if } (x==0) \#s \text{ else } \#t \phi \leftrightarrow \langle l : \#t \phi \rangle \rangle
\]

Generation of Proof Obligations to Ensure the Soundness of Taclets – p.15/20
Example of Proof Obligation (2)

\[(x \teq 0 \land
\langle l : \textbf{if } (x==0) \textbf{ else } \neg t \rangle \phi \leftrightarrow \langle l : \neg s \rangle \phi \rangle
\lor \neg (x = 0) \land
\langle l : \textbf{if } (x==0) \textbf{ else } \neg t \rangle \phi \leftrightarrow \langle l : s \rangle \phi \rangle\)
Example of Proof Obligation (2)

\[(x \equiv 0 \land \langle l : \text{if } (x==0) \#s \text{ else } \#t \rangle \phi \leftrightarrow \langle l : \#s \rangle \phi)\]
\[\lor (\neg (x \equiv 0) \land \langle l : \text{if } (x==0) \#s \text{ else } \#t \rangle \phi \leftrightarrow \langle l : \#t \rangle \phi)\]

Proof obligation of \(t_2\):

\[(x \equiv 0 \land \langle l : \text{if } (x==0) \beta_1 \text{ else } \beta_2 \rangle p_{Sk}(x) \leftrightarrow \langle l : \beta_1 \rangle p_{Sk}(x))\]
\[\lor (\neg (x \equiv 0) \land \langle l : \text{if } (x==0) \beta_1 \text{ else } \beta_2 \rangle p_{Sk}(x) \leftrightarrow \langle l : \beta_2 \rangle p_{Sk}(x))\]
Example of Proof Obligation (2)

Proof obligation of $t_2$:

$$
\left( x \div 0 \land

\langle l : \textbf{if} \ (x==0) \ \beta_1 \ \textbf{else} \ \beta_2 \rangle p_{Sk}(x) \leftrightarrow \langle l : \beta_1 \rangle p_{Sk}(x) \rangle

\lor

\langle l : \textbf{if} \ (x==0) \ \beta_1 \ \textbf{else} \ \beta_2 \rangle p_{Sk}(x) \leftrightarrow \langle l : \beta_2 \rangle p_{Sk}(x) \rangle
\right)
$$
Example of Proof Obligation (2)

Proof obligation of $t_2$:

\[
(x \div 0 \land
(\langle l : \textbf{if } (x==0) \beta_1 \textbf{ else } \beta_2 \rangle p_{sk}(x) \leftrightarrow \langle l : \beta_1 \rangle p_{sk}(x)))
\]

$\vee (\neg (x \div 0) \land
(\langle l : \textbf{if } (x==0) \beta_1 \textbf{ else } \beta_2 \rangle p_{sk}(x) \leftrightarrow \langle l : \beta_2 \rangle p_{sk}(x)))$

$\beta_1 = s_{sk}(x, t_#s, d_#s; \textbf{break } l; \textbf{throw } t_#s)$;

$\beta_2 = t_{sk}(x, t_#t, d_#t; \textbf{break } l; \textbf{throw } t_#t)$;
Summary

- The presented approach ...
The presented approach . . .
- treats first-order taclets completely

Future work:
- Support more JavaCardDL schema variables
- Consider some special characteristics of KeY, e.g. untyped schema variables
The presented approach ... 
  - treats first-order taclets completely
  - handles the most important kinds of JavaCardDL schema variables

Future work:
- Support more JavaCardDL schema variables
- Consider some special characteristics of KeY, e.g. untyped schema variables
The presented approach . . .
  • treats first-order taclets completely
  • handles the most important kinds of JavaCardDL schema variables
  • is fully implemented

Future work:
  • Support more JavaCardDL schema variables
  • Consider some special characteristics of KeY, e.g. untyped schema variables
Summary

- The presented approach ...  
  - treats first-order taclets completely  
  - handles the most important kinds of JavaCardDL schema variables  
  - is fully implemented  
- Future work:
Summary

- The presented approach . . .
  - treats first-order taclets completely
  - handles the most important kinds of JavaCardDL schema variables
  - is fully implemented
- Future work:
  - Support more JavaCardDL schema variables
Summary

• The presented approach . . .
  • treats first-order taclets completely
  • handles the most important kinds of JavaCardDL schema variables
  • is fully implemented

• Future work:
  • Support more JavaCardDL schema variables
  • Consider some special characteristics of KeY, e.g. untyped schema variables