

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

Introduction to Taclets (2)

- Taclet in concrete syntax:

```
if (ifseq) find (f)  
  replacewith (rw1) add (add1);  
  ⋮  
  replacewith (rwk) add (addk)
```


Introduction to Taclets (2)

- Taclet in concrete syntax:

```
if (ifseq) find (f)  
  replacewith (rw1) add (add1);  
  ⋮  
  replacewith (rwk) add (addk)
```

- For instance: Modus ponens

```
if ( $\phi \vdash$  ) find ( $\vdash \phi \rightarrow \psi$ )  
  replacewith ( $\vdash \psi$ )
```

Schema Variables

- For first-order logic:

Schema Variables

- For first-order logic:
 - Terms

Schema Variables

- For first-order logic:
 - Terms
 - Formulas

Schema Variables

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

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

$$\frac{P_1 \quad P_2 \quad \cdots \quad P_k}{Q}$$

$$P_1, P_2, \dots, P_k \text{ valid} \implies Q \text{ valid}$$

-
-
-

Concept to Ensure Soundness of Tactlets

Method introduced by Elmar Habermalz:

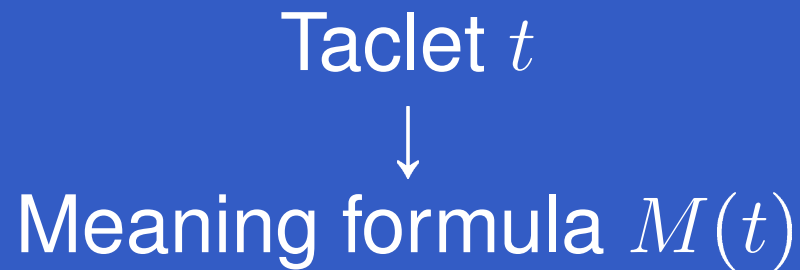
Concept to Ensure Soundness of Taclets

Method introduced by Elmar Habermalz:

Taclet t

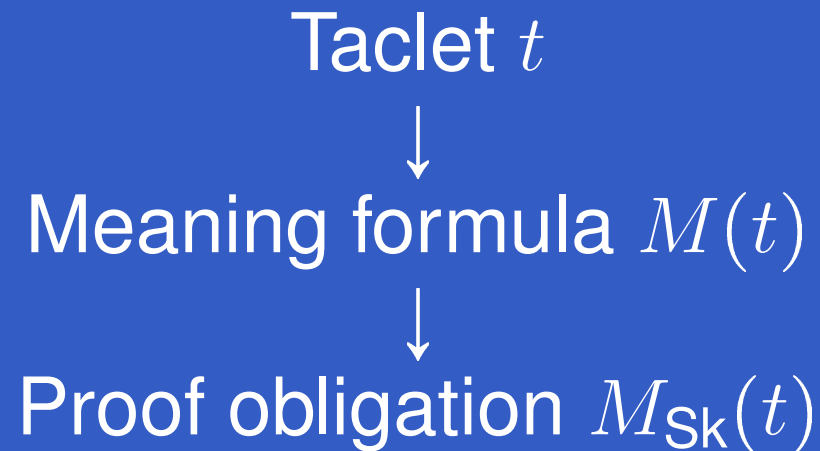
Concept to Ensure Soundness of Taclets

Method introduced by Elmar Habermalz:



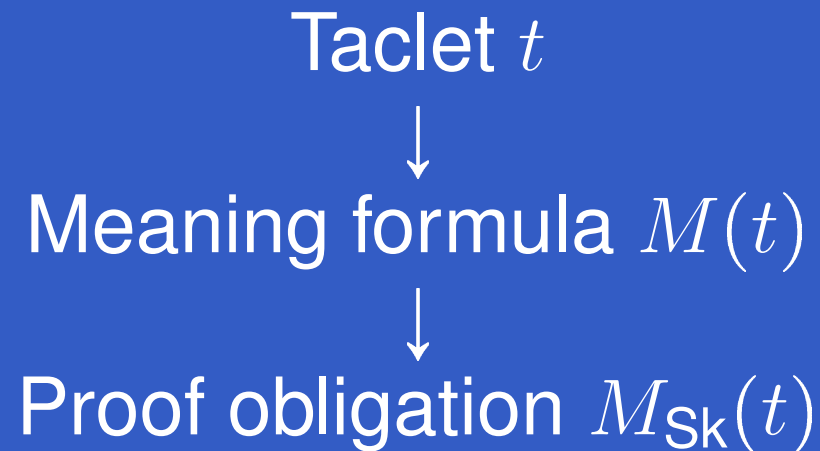
Concept to Ensure Soundness of Taclets

Method introduced by Elmar Habermalz:



Concept to Ensure Soundness of Taclets

Method introduced by Elmar Habermalz:



$M_{Sk}(t)$ valid $\iff t$ sound

Example of Meaning Formula

- Taclet t_1 exchanging quantifiers:

find ($\vdash \forall y. \forall x. \phi$)

replacewith ($\vdash \forall x. \forall y. \phi$)

Example of Meaning Formula

- Taclet t_1 exchanging quantifiers:

find ($\vdash \forall y. \forall x. \phi$)

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:

```
find(<l: if (x==0) #s else #t>ϕ)
```

```
  replacewith(<l: #s >ϕ)  add(x ≐ 0 ⊢ );
```

```
  replacewith(<l: #t >ϕ)  add( ⊢ x ≐ 0)
```

Example of Meaning Formula (2)

`find(<l: if (x==0) #s else #t> ϕ)`

`replacewith(<l: #s> ϕ) add(x \doteq 0 \vdash);`

`replacewith(<l: #t> ϕ) add(\vdash x \doteq 0)`

Example of Meaning Formula (2)

```
find(<l: if (x==0) #s else #t>ϕ)
  replacewith(<l: #s >ϕ)   add(x ≐ 0 ⊢ );
  replacewith(<l: #t >ϕ)   add( ⊢ x ≐ 0)
```

$M(t_2) =$

$$\begin{aligned} & (x \doteq 0 \wedge \\ & \quad (\langle l: \mathbf{if} (x==0) \#s \mathbf{else} \#t \rangle \phi \leftrightarrow \langle l: \#s \rangle \phi)) \\ \vee & (\neg(x \doteq 0) \wedge \\ & \quad (\langle l: \mathbf{if} (x==0) \#s \mathbf{else} \#t \rangle \phi \leftrightarrow \langle l: \#t \rangle \phi)) \end{aligned}$$

Taclet Proof Obligations

- Schema variables of meaning formulas are replaced with skolem symbols

Taclet Proof Obligations

- Schema variables of meaning formulas are replaced with skolem symbols

SVs for terms → 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

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

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:

$$M_{\text{Sk}}(t_1) = \forall u.\forall v.p_{\text{Sk}}(u, v) \rightarrow \forall v.\forall u.p_{\text{Sk}}(u, v)$$

Example of Proof Obligation (2)

- Meaning formula of t_2 is

$$\begin{aligned} & (x \doteq 0 \wedge \\ & \quad (\langle l : \mathbf{if} (x==0) \#s \mathbf{else} \#t \rangle \phi \leftrightarrow \langle l : \#s \rangle \phi)) \\ \vee & (\neg(x \doteq 0) \wedge \\ & \quad (\langle l : \mathbf{if} (x==0) \#s \mathbf{else} \#t \rangle \phi \leftrightarrow \langle l : \#t \rangle \phi)) \end{aligned}$$

Example of Proof Obligation (2)

$(x \doteq 0 \wedge$

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

$\vee (\neg(x \doteq 0) \wedge$

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

Example of Proof Obligation (2)

$$\begin{aligned} & (x \doteq 0 \wedge \\ & \quad (\langle l : \mathbf{if} (x==0) \#s \mathbf{else} \#t \rangle \phi \leftrightarrow \langle l : \#s \rangle \phi)) \\ \vee & (\neg(x \doteq 0) \wedge \\ & \quad (\langle l : \mathbf{if} (x==0) \#s \mathbf{else} \#t \rangle \phi \leftrightarrow \langle l : \#t \rangle \phi)) \end{aligned}$$

Proof obligation of t_2 :

$$\begin{aligned} & (x \doteq 0 \wedge \\ & \quad (\langle l : \mathbf{if} (x==0) \beta_1 \mathbf{else} \beta_2 \rangle p_{Sk}(x) \leftrightarrow \langle l : \beta_1 \rangle p_{Sk}(x))) \\ \vee & (\neg(x \doteq 0) \wedge \\ & \quad (\langle l : \mathbf{if} (x==0) \beta_1 \mathbf{else} \beta_2 \rangle p_{Sk}(x) \leftrightarrow \langle l : \beta_2 \rangle p_{Sk}(x))) \end{aligned}$$

Example of Proof Obligation (2)

Proof obligation of t_2 :

$$(x \doteq 0 \wedge$$

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

$$\vee (\neg(x \doteq 0) \wedge$$

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

Example of Proof Obligation (2)

Proof obligation of t_2 :

$$\begin{aligned} & (x \doteq 0 \wedge \\ & \quad (\langle l : \mathbf{if} (x==0) \beta_1 \mathbf{else} \beta_2 \rangle p_{Sk}(x) \leftrightarrow \langle l : \beta_1 \rangle p_{Sk}(x))) \\ \vee & (\neg(x \doteq 0) \wedge \\ & \quad (\langle l : \mathbf{if} (x==0) \beta_1 \mathbf{else} \beta_2 \rangle p_{Sk}(x) \leftrightarrow \langle l : \beta_2 \rangle p_{Sk}(x))) \end{aligned}$$

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

$\beta_2 = t_{Sk}(x, t_{\#t}, d_{\#t}; \mathbf{break} \ l; \mathbf{throw} \ t_{\#t});$

Summary

- The presented approach . . .

Summary

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

Summary

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

Summary

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

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