

# Theorembeweiserpraktikum

## Anwendungen in der Sprachtechnologie

<http://pp.info.uni-karlsruhe.de/lehre/SS2010/tba/>  
Daniel Wasserrab

IPD Snelting, Lehrstuhl Programmierparadigmen



# Teil XI

## Strukturierte Beweise mittels Isar

# Was ist Isar?

**apply**-Skripten für externen Leser völlig unverständlich

# Was ist Isar?

**apply**-Skripten für externen Leser völlig unverständlich

Lösung: **Isar**!

# Was ist Isar?

**apply**-Skripten für externen Leser völlig unverständlich

Lösung: **Isar**!

- Vater des Gedanken: Mizar
- bessere Strukturierung durch Herleitung von “Zwischenaussagen”
- aussagekräftigere Schlüsselwörter (ähnlich mathematischer Notation)
- (gut geschriebener) Beweis verständlich ohne aktuellen Beweiszustand
- einfachere Anpassung bei Änderungen

# Aufbau eines einfachen Beispiels

```
lemma "A  $\longrightarrow$  A"  
proof(rule impI)  
  assume "A"  
  from 'A' show "A" by(rule a)  
qed
```

# Aufbau eines einfachen Beispiels

**lemma** "A  $\longrightarrow$  A"

**proof**(rule impI)

**assume** "A"

**from** 'A' **show** "A" **by**(rule a)

**qed**

allgemeiner Aufbau:

- Lemmadefinition wie gewohnt
- dann **proof**, gefolgt von Beweismethode
- Prämissen des aktuellen subgoals mit **assume** dargestellt (können auch Namen erhalten)
- Ziel des aktuellen subgoals mittels **show** ausgedrückt, danach folgt Beweis der Aussage
- **qed** als "schliessende Klammer" für jedes **proof**

# Wie beweise ich in Isar?

Ziel in Isar: für jeden Beweis nur **die dafür nötigen Prämissen** verwenden  
dafür: Zwischenaussagen beweisen, bilden Prämissen für nächste  
Zwischenaussage oder das ganze Beweisziel  
Zwischenaussage nach Schlüsselwort **have**, danach evtl. Name,  
danach Aussage, gefolgt von Beweis  
dieser Beweis mittels **apply**-Skript oder Isar-**proof**

# Wie beweise ich in Isar?

Ziel in Isar: für jeden Beweis nur **die dafür nötigen Prämissen** verwenden  
dafür: Zwischenaussagen beweisen, bilden Prämissen für nächste

Zwischenaussage oder das ganze Beweisziel

Zwischenaussage nach Schlüsselwort **have**, danach evtl. Name,  
danach Aussage, gefolgt von Beweis

dieser Beweis mittels **apply**-Skript oder Isar-**proof**

Prämissen eines Beweises einer Zwischenaussage bzw. des Beweisziels

- aus Annahmen (Aussagen nach **assume**)
- aus bereits bewiesenen Zwischenaussagen (Aussagen nach **have**)

werden vor der Beweisaussage nach Schlüsselwort **from** aufgesammelt  
mittels Name oder Aussage in schrägen Hochkommata ``

diese beiden Zitationsarten können beliebig gemischt werden

# Wie beweise ich in Isar?

Beispiel:

**lemma**  $"A \wedge B \longrightarrow B \wedge A"$

**proof**(*rule impI*)

**assume**  $ab: "A \wedge B"$

**from**  $ab$  **have**  $a: "A"$  **by**  $-(erule conjE)$

**from**  $\`A \wedge B\`$  **have**  $"B"$  **by**  $-(erule conjE)$

**from**  $\`B\` a$  **show**  $"B \wedge A"$  **by**(*rule conjI*)

**qed**

# Wie beweise ich in Isar?

Beispiel:

**lemma** "A  $\wedge$  B  $\longrightarrow$  B  $\wedge$  A"

**proof**(rule impI)

**assume** ab: "A  $\wedge$  B"

**from** ab **have** a:"A" **by** -(erule conjE)

**from** `A  $\wedge$  B` **have** "B" **by** -(erule conjE)

**from** `B` a **show** "B  $\wedge$  A" **by**(rule conjI)

**qed**

hier werden Namen und direkte Zitationen fröhlich gemischt,  
in sauberem Beweis besser

- direkt zitieren bei kurzen Aussagen,
- Namen für längere Aussagen

direkt zitieren ist lesbarer, Namen anpassbarer bei Änderungen

# Strukturieren des Beweises

aktuell gezeigte oder angenommene Aussage mittels *this* verfügbar:

**lemma** "A  $\rightarrow$  A"

**proof**(*rule impI*)

**assume** "A"

**from** *this* **show** "A" *by assumption*

**qed**

also Beweis möglichst so strukturieren, dass jede Aussage auf der vorherigen aufbaut

aktuell gezeigte oder angenommene Aussage mittels *this* verfügbar:

**lemma** " $A \rightarrow A$ "

**proof**(*rule impI*)

**assume** " $A$ "

**from** *this* **show** " $A$ " *by assumption*

**qed**

also Beweis möglichst so strukturieren, dass jede Aussage auf der vorherigen aufbaut

Syntaktische Abkürzungen:

- **from** *this*  $\equiv$  **then** (nur diese eine Annahme!)
- **from**  $a$   $b$  *this*  $\equiv$  **with**  $a$   $b$  (mehrere Annahmen)
- **from** *this* **show**  $\equiv$  **then show**  $\equiv$  **thus**
- **from** *this* **have**  $\equiv$  **then have**  $\equiv$  **hence**

## proof ohne Regel

wenn nach **proof** keine Methode spezifiziert, wird (falls möglich) sinnvolle erste (Introduktions- oder Eliminations-) Regel angewandt  
kann zu unlösbaren subgoals führen, z.B. bei Disjunktion in Konklusion

## proof ohne Regel

wenn nach **proof** keine Methode spezifiziert, wird (falls möglich) sinnvolle erste (Introduktions- oder Eliminations-) Regel angewandt  
 kann zu unlösbaren subgoals führen, z.B. bei Disjunktion in Konklusion

Beispiel:

**lemma** "[ $A \vee B; A \rightarrow Q$ ]  $\implies (B \rightarrow Q) \rightarrow Q$ "

**proof** — verwendet implizit (*rule impI*)

**assume** " $A \vee B$ " and " $A \rightarrow Q$ " and " $B \rightarrow Q$ "

**from** `A  $\vee$  B` **show** " $Q$ "

**proof** — verwendet implizit (*erule disjE*)

**assume** " $A$ "

**from** `A` `A  $\rightarrow$  Q` **show** ?thesis **by** simp

**next**

**assume** " $B$ "

**from** `B` `B  $\rightarrow$  Q` **show** ?thesis **by** simp

**qed**

**qed**

mit ?thesis wird aktuelles Ziel abgekürzt, hier  $Q$

# Methoden für proof

**proof** entweder explizit übergebene Introduktions- oder Eliminationsregel  
oder implizit automatisch gewählte (wenn Beweismethode weggelassen)

**proof** entweder explizit übergebene Introduktions- oder Eliminationsregel oder implizit automatisch gewählte (wenn Beweismethode weggelassen)

weitere Möglichkeiten:

- keine Beweismethode passend oder Automatismus nicht gewollt:  
**proof** -: Beweisziel unverändert
- Fallunterscheidung durch **proof**(*cases P*)
- strukturelle Induktion über Datentyp *D* durch **proof**(*induct D*)
- Regelinduktion (für **fun** und **inductive**) analog  
**proof**(*induct rule:bla.induct*)

bitte kein **proof simp** oder **proof auto**!

# Methoden für proof

Fallunterscheidung und andere Methoden erzeugen mehrere subgoals wenn erstes bewiesen, Übergang zum nächsten mittels **next** (statt **qed**)

Fallunterscheidung und andere Methoden erzeugen mehrere subgoals wenn erstes bewiesen, Übergang zum nächsten mittels **next** (statt **qed**)

**lemma** " $(A \longrightarrow B) \wedge (\neg A \longrightarrow B) \longrightarrow B$ "

**proof**

**assume** " $(A \longrightarrow B) \wedge (\neg A \longrightarrow B)$ "

**hence** " $A \longrightarrow B$ " **and** " $\neg A \longrightarrow B$ " **by** *-(erule conjE/assumption)+*

**show** " $B$ "

**proof**(cases A)

**assume** " $A$ "

**with**  $A \longrightarrow B$  **show** *?thesis* **by**(rule mp)

**next**

**assume** " $\neg A$ "

**with**  $\neg A \longrightarrow B$  **show** *?thesis* **by**(rule mp)

**qed**

**qed**

# Lemma mit Prämissen und Konklusion

Lemma besteht aus Prämissen und Konklusion, dann

- Liste der Prämissen nach Schlüsselwort **assumes**, evtl. mit Namen und mit **and** getrennt
- Konklusion nach **shows**
- Lemmaname kann entweder wie bisher vor allen **assumes** angegeben werden oder direkt vor **shows**

Prämissen können dann im Beweisskript zitiert werden

# Lemma mit Prämissen und Konklusion

Lemma besteht aus Prämissen und Konklusion, dann

- Liste der Prämissen nach Schlüsselwort **assumes**, evtl. mit Namen und mit **and** getrennt
- Konklusion nach **shows**
- Lemmaname kann entweder wie bisher vor allen **assumes** angegeben werden oder direkt vor **shows**

Prämissen können dann im Beweisskript zitiert werden

so wird aus **lemma** *bla*: " $\llbracket P; Q \rrbracket \implies R$ "

**lemma assumes** "P" **and** "Q" **shows** *bla*: "R"

# Lemma mit Prämissen und Konklusion

**Vorsicht!** Induktionsregeln o.ä. benötigen eine oder mehrere Prämissen, um die Regel anwenden zu können  
in dieser Schreibform: angeben, welche Prämissen für Regel nötig sind  
Syntax: vor **proof** hinter Schlüsselwort **using** Namen der benötigten Prämissen

# Lemma mit Prämissen und Konklusion

**Vorsicht!** Induktionsregeln o.ä. benötigen eine oder mehrere Prämissen, um die Regel anwenden zu können

in dieser Schreibform: angeben, welche Prämissen für Regel nötig sind

Syntax: vor **proof** hinter Schlüsselwort **using** Namen der benötigten Prämissen

Beispiel:

**lemma assumes** *rev: "rev xs = ys"* **shows** *"rev ys = xs"*

**proof**(*induct xs*)

liefert unbeweisbare subgoals, da nur über *xs* in Konklusion induziert wird

# Lemma mit Prämissen und Konklusion

**Vorsicht!** Induktionsregeln o.ä. benötigen eine oder mehrere Prämissen, um die Regel anwenden zu können

in dieser Schreibform: angeben, welche Prämissen für Regel nötig sind

Syntax: vor **proof** hinter Schlüsselwort **using** Namen der benötigten Prämissen

Beispiel:

**lemma assumes** *rev: "rev xs = ys"* **shows** *"rev ys = xs"*

**using** *rev*

**proof**(*induct xs*)

Induktion über *xs* in Prämisse und Konklusion, damit subgoals beweisbar

# Allquantoren in zu zeigenden Aussagen

Wie gehe ich mit Parametern um?

Beispiel:

**lemma assumes** " $\neg(\exists x. \neg P x)$ " **shows** " $\forall x. P x$ "

**proof**

nach Anwendung von **proof** (implizit *allI*) subgoal:  $\wedge x. P x$   
doch **show** " $\wedge x. P x$ " funktioniert nicht!

# Allquantoren in zu zeigenden Aussagen

Wie gehe ich mit Parametern um?

Beispiel:

**lemma assumes** " $\neg(\exists x. \neg P x)$ " **shows** " $\forall x. P x$ "  
**proof**

nach Anwendung von **proof** (implizit *allI*) subgoal:  $\wedge x. P x$   
doch **show** " $\wedge x. P x$ " funktioniert nicht!

Parameter müssen zu Beginn des Beweises "gefixt" werden:  
 $\wedge a b c. P$  wird zu **fix**  $a b c$  und **show**  $P$

# Allquantoren in zu zeigenden Aussagen

Wie gehe ich mit Parametern um?

Beispiel:

**lemma assumes** " $\neg(\exists x. \neg P x)$ " **shows** " $\forall x. P x$ "  
**proof**

nach Anwendung von **proof** (implizit *allI*) subgoal:  $\bigwedge x. P x$   
doch **show** " $\bigwedge x. P x$ " funktioniert nicht!

Parameter müssen zu Beginn des Beweises "gefixt" werden:

$\bigwedge a b c. P$  wird zu **fix**  $a b c$  und **show**  $P$

Unser Beispiel:

**lemma assumes** " $\neg(\exists x. \neg P x)$ " **shows** " $\forall x. P x$ "  
**proof**  
    **fix**  $x$   
    **show** " $P x$ " ...  
**qed**

# Repräsentanten

Um aus Existenzquantoren Repräsentanten zu erhalten, **obtain**

Syntax: **obtain** *Repräsentant* **where** *Aussage Beweis*

Beispiel:

**lemma** " $\exists x. P x \implies \neg (\forall x. \neg P x)$ "

**proof** -

**assume** *ex*: " $\exists x. P x$ "

**from** *ex* **obtain** *z* **where** " $P z$ " **by** *blast*

    ...

Um aus Existenzquantoren Repräsentanten zu erhalten, **obtain**

Syntax: **obtain** *Repräsentant* **where** *Aussage Beweis*

Beispiel:

**lemma** " $\exists x. P\ x \implies \neg (\forall x. \neg P\ x)$ "

**proof** -

**assume** *ex*: " $\exists x. P\ x$ "

**from** *ex* **obtain** *z* **where** " $P\ z$ " **by** *blast*

...

- Repräsentantename kann beliebig gewählt sein (nicht vorhandene Variable!)
- Aussage nach **where** kann beliebigen Namen erhalten
- auch aus Allquantor **obtain** möglich (da Typen in Isabelle immer nichtleer)

# obtain-Beweise

Was genau passiert bei den **obtain**-Beweisen?

Isabelle baut aus Zeile **obtain**  $x$  **where** " $P\ x$ "

Beweisziel  $(\wedge x. P\ x \implies thesis) \implies thesis$  (*thesis* beliebige Aussage)

# obtain-Beweise

Was genau passiert bei den **obtain**-Beweisen?

Isabelle baut aus Zeile **obtain**  $x$  **where** " $P\ x$ "

Beweisziel  $(\wedge x. P\ x \implies thesis) \implies thesis$  ( $thesis$  beliebige Aussage)

Also (mittels geeigneter Annahmen) linke Seite der

" $P\ x \implies thesis$ "-Metaimplikation zeigen für beliebige  $x$

Was genau passiert bei den **obtain**-Beweisen?

Isabelle baut aus Zeile **obtain**  $x$  **where** " $P\ x$ "

Beweisziel  $(\bigwedge x. P\ x \implies thesis) \implies thesis$  ( $thesis$  beliebige Aussage)

Also (mittels geeigneter Annahmen) linke Seite der

" $P\ x \implies thesis$ "-Metaimplikation zeigen für beliebige  $x$

Trotzdem **Keine Panik!**

- Wenn nach **where** exakt Aussage einer Existenzaussage, Beweis einfach mittel *blast*
- auch sonst meist Beweis mittels *blast*, *auto* oder *fastsimp*
- manchmal evtl. noch Fallunterscheidung nötig (z.B. Variable ist Tupel, Bestimmen der Tuppeleinträge)