Theorembeweiser und ihre Anwendungen

Prof. Dr.-Ing. Gregor Snelting
Dipl.-Inf. Univ. Daniel Wasserrab

Lehrstuhl Programmierparadigmen
IPD Snelting
Universität Karlsruhe (TH)

Ziel der Vorlesung

- Wesen der Theorembeweiser
- Einblick in aktuelle Forschungsthemen mit Theorembeweiserbezug
- (praktische) Arbeit mit Theorembeweiser

Ziel der Vorlesung

- Wesen der Theorembeweiser
- Einblick in aktuelle Forschungsthemen mit Theorembeweiserbezug
- (praktische) Arbeit mit Theorembeweiser

Diese Vorlesung beinhaltet nicht:

- Einführung in funktionale Programmierung
- Logikprogrammierung (Prolog)
- (Tiefen der) Logik, Tableaux-, Resolutionskalkül, etc.
- interne Funktionsweise von Theorembeweisern
- Wie programmiere ich einen Theorembeweiser?

Organisatorisches

Realisierung: Vorlesung mit Rechnerübung unter Anleitung

Diplom-/Masterstudiengang Informatik, Vertiefungsfach
Vertiefungsgebiete:

- 01 Theoretische Grundlagen
- 06 Softwaretechnik und Übersetzerbau

Vorlesung: Donnerstag, 8.45 - 9.30, R -143

Forschungsorientiert, Fokus auf aktuelle Arbeiten

`http://pp.info.uni-karlsruhe.de/lehre/SS2009/tba/index.php`

Rechnerübung unter Anleitung + Übung: Dienstag, 11.30 - 13.00, R -143

Grundlagen des Beweisassistenten Isabelle/HOL
(und der Benutzeroberfläche Isar)
Eigenständiges Lösen der Aufgaben unter Anleitung

`http://pp.info.uni-karlsruhe.de/lehre/SS2009/tba/uebung.php`

Vorlesung

Behandelte Fragen:

- Was ist ein Theorembeweiser?
- Welche Theorembeweiser gibt es? (Auswahl!)
- Worin unterscheiden sich verschiedene Theorembeweiser?
- Was kann ein Theorembeweiser leisten (und was nicht)?
- Wozu verwendet man Theorembeweiser? (Auswahl!)

Vorlesung

Behandelte Fragen:

- Was ist ein Theorembeweiser?
- Welche Theorembeweiser gibt es? (Auswahl!)
- Worin unterscheiden sich verschiedene Theorembeweiser?
- Was kann ein Theorembeweiser leisten (und was nicht)?
- Wozu verwendet man Theorembeweiser? (Auswahl!)

Fokus auf den letzten Punkt

Vorlesung

Einsatz von Theorembeweisern in der aktuellen Forschung an Beispielen:

- Anwendungen in der Mathematik
- Verifikation von kryptographischen Protokollen
- formale Semantiken und Typsicherheit
- Typbasierte Informationsflußkontrolle
- Verifikation eines Compiler

evtl. weitere Themen

Keine Literatur zur Vorlesung
Vorlesungsfolien und entsprechende Paper (für Interessierte) werden vorher auf Vorlesungsseite zum Download bereit gestellt

Vorlesung

Einsatz von Theorembeweisern in der aktuellen Forschung an Beispielen:

- Anwendungen in der Mathematik
- Verifikation von kryptographischen Protokollen
- formale Semantiken und Typsicherheit
- Typbasierte Informationsflußkontrolle
- Verifikation eines Compiler

evtl. weitere Themen

Keine Literatur zur Vorlesung
Vorlesungsfolien und entsprechende Paper (für Interessierte) werden vorher auf Vorlesungsseite zum Download bereit gestellt

Übung

Erste Schritte mit einem ausgewählten Theorembeweiser: Isabelle/HOL
Kennenlernen der Oberfläche Isar (verständliche Notation)
Behandelte Themen:

- Regeln, Deduktion, Quantoren
- Verwendung automatischer Beweistaktiken
- Funktionale Programmierung
  - Datentypen
  - primitive Rekursion
  - strukturelle Induktion
- Prädikate, Mengen und Relationen
  - Induktive Prädikate und Mengen
  - Regelinduktion
  - Reflexive transitive Hülle

Übung

Erste Schritte mit einem ausgewählten Theorembeweiser: Isabelle/HOL
Kennenlernen der Oberfläche Isar (verständliche Notation)
Behandelte Themen:

- Regeln, Deduktion, Quantoren
- Verwendung automatischer Beweistaktiken
- Funktionale Programmierung
  - Datentypen
  - primitive Rekursion
  - strukturelle Induktion
- Prädikate, Mengen und Relationen
  - Induktive Prädikate und Mengen
  - Regelinduktion
  - Reflexive transitive Hülle

Übung

Erste Schritte mit einem ausgewählten Theorembeweiser: Isabelle/HOL
Kennenlernen der Oberfläche Isar (verständliche Notation)
Behandelte Themen:

- Regeln, Deduktion, Quantoren
- Verwendung automatischer Beweistaktiken
- Funktionale Programmierung
  - Datentypen
  - primitive Rekursion
  - strukturelle Induktion
- Prädikate, Mengen und Relationen
  - Induktive Prädikate und Mengen
  - Regelinduktion
  - Reflexive transitive Hülle

Übung

Erste Schritte mit einem ausgewählten Theorembeweiser: Isabelle/HOL
Kennenlernen der Oberfläche Isar (verständliche Notation)
Behandelte Themen:

- Regeln, Deduktion, Quantoren
- Verwendung automatischer Beweistaktiken
- Funktionale Programmierung
  - Datentypen
  - primitive Rekursion
  - strukturelle Induktion
- Prädikate, Mengen und Relationen
  - Induktive Prädikate und Mengen
  - Regelinduktion
  - Reflexive transitive Hülle

Übung

Erste Schritte mit einem ausgewählten Theorembeweiser: Isabelle/HOL
Kennenlernen der Oberfläche Isar (verständliche Notation)
Behandelte Themen:

- Regeln, Deduktion, Quantoren
- Verwendung automatischer Beweistaktiken
- Funktionale Programmierung
  - Datentypen
  - primitive Rekursion
  - strukturelle Induktion
- Prädikate, Mengen und Relationen
  - Induktive Prädikate und Mengen
  - Regelinduktion
  - Reflexive transitive Hülle

Rechnerübung unter Anleitung

- zu Beginn jeder Übung Folien zum aktuellen Thema
  Folien Montag vor Übung zum Download auf Übungsseite
- dann selbstständig Bearbeitung der Übungen am Rechner
  Isabelle-Rahmen verfügbar auf Übungsseite
  Hilfe und Unterstützung durch anwesenden Betreuer
- falls Zeit der Übung nicht ausreicht, sollten Aufgaben eigenständig fertig bearbeitet werden
- jede Woche neues Thema

Literatur Übung

T. Nipkow, L. C. Paulson and M. Wenzel.
Isabelle/HOL - The Tutorial.
Springer, 2008.
Das Buch ist leider ziemlich veraltet, Download aktuelle Version:
`http://isabelle.in.tum.de/dist/Isabelle/doc/tutorial.pdf`

D. J. Velleman.
How to Prove it: A Structural Approach.
Cambridge University Press, 1996.
Allgemeine Einführung in Beweistheorie

Teil I

Was ist ein Theorembeweiser?

Erstmal ganz abstrakt...

Ein Theorembeweiser beweist Aussagen über formale Strukturen durch Anwendung von Regeln.

Etwas genauer...

Ein Theorembeweiser beweist Aussagen über *formale Strukturen* durch Anwendung von Regeln.

- Typen und Datentypen (natürliche Zahlen, Listen, Paare, . . .)
- Mengen, Relationen, Funktionen
- funktionale Programmierung ermöglicht selbstdefinierte Strukturen (durch Rekursion, Fallunterscheidung etc.)
- definiert im jeweiligen System!

Etwas genauer...

Ein Theorembeweiser beweist Aussagen über *formale Strukturen* durch Anwendung von Regeln.

- Typen und Datentypen (natürliche Zahlen, Listen, Paare, . . .)
- Mengen, Relationen, Funktionen
- funktionale Programmierung ermöglicht selbstdefinierte Strukturen (durch Rekursion, Fallunterscheidung etc.)
- definiert im jeweiligen System!

Etwas genauer...

Ein Theorembeweiser beweist Aussagen über *formale Strukturen* durch Anwendung von Regeln.

- Typen und Datentypen (natürliche Zahlen, Listen, Paare, . . .)
- Mengen, Relationen, Funktionen
- funktionale Programmierung ermöglicht selbstdefinierte Strukturen (durch Rekursion, Fallunterscheidung etc.)
- definiert im jeweiligen System!

Etwas genauer...

Ein Theorembeweiser beweist Aussagen über *formale Strukturen* durch Anwendung von Regeln.

- Typen und Datentypen (natürliche Zahlen, Listen, Paare, . . .)
- Mengen, Relationen, Funktionen
- funktionale Programmierung ermöglicht selbstdefinierte Strukturen (durch Rekursion, Fallunterscheidung etc.)
- definiert im jeweiligen System!

Etwas genauer...

Ein Theorembeweiser beweist Aussagen über *formale Strukturen* durch Anwendung von Regeln.

- Typen und Datentypen (natürliche Zahlen, Listen, Paare, . . .)
- Mengen, Relationen, Funktionen
- funktionale Programmierung ermöglicht selbstdefinierte Strukturen (durch Rekursion, Fallunterscheidung etc.)
- definiert im jeweiligen System!

Etwas genauer...

Ein Theorembeweiser *beweist Aussagen* über formale Strukturen durch Anwendung von Regeln.

automatisch:

prozedural:

deklarativ:

Etwas genauer...

Ein Theorembeweiser *beweist Aussagen* über formale Strukturen durch Anwendung von Regeln.

automatisch:

- Theorembeweiser versucht Ziel eigenständig zu lösen
- bei Nichtgelingen Meldung, woran gescheitert und Abbruch
- Hilfslemmas zeigen und Beweisprozess hinzufügen
- nochmals versuchen, Ziel zu zeigen

prozedural:

deklarativ:

Etwas genauer...

Ein Theorembeweiser *beweist Aussagen* über formale Strukturen durch Anwendung von Regeln.

automatisch:

prozedural:

- Taktiken für bestimmte automatisierte Prozesse
- können durch vorher gezeigte Hilfslemmas erweitert werden
- Beweisprozess wird nicht abgebrochen falls erfolglos, sondern an den Benutzer übergeben
- mittels Beweisskripten 'Dirigieren' der Schlussfolgerung

deklarativ:

Etwas genauer...

Ein Theorembeweiser *beweist Aussagen* über formale Strukturen durch Anwendung von Regeln.

automatisch:

prozedural:

deklarativ:

- Benutzer schreibt kompletten Beweis
- System prüft den Beweis
- bricht ab, falls Schlussfolgerung nicht korrekt

Etwas genauer...

Ein Theorembeweiser beweist Aussagen über formale Strukturen durch *Anwendung von Regeln*.

- Unifikation und Substitution
- Simplifikation
- (natürliche) Deduktion inkl. Quantoren
- Induktion (natürlich, wohlgeformt, strukturell, Regel-Ind., . . .)

Etwas genauer...

Ein Theorembeweiser beweist Aussagen über formale Strukturen durch *Anwendung von Regeln*.

- Unifikation und Substitution
- Simplifikation
- (natürliche) Deduktion inkl. Quantoren
- Induktion (natürlich, wohlgeformt, strukturell, Regel-Ind., . . .)

Etwas genauer...

Ein Theorembeweiser beweist Aussagen über formale Strukturen durch *Anwendung von Regeln*.

- Unifikation und Substitution
- Simplifikation
- (natürliche) Deduktion inkl. Quantoren
- Induktion (natürlich, wohlgeformt, strukturell, Regel-Ind., . . .)

Etwas genauer...

Ein Theorembeweiser beweist Aussagen über formale Strukturen durch *Anwendung von Regeln*.

- Unifikation und Substitution
- Simplifikation
- (natürliche) Deduktion inkl. Quantoren
- Induktion (natürlich, wohlgeformt, strukturell, Regel-Ind., . . .)

Etwas genauer...

Ein Theorembeweiser beweist Aussagen über formale Strukturen durch *Anwendung von Regeln*.

- Unifikation und Substitution
- Simplifikation
- (natürliche) Deduktion inkl. Quantoren
- Induktion (natürlich, wohlgeformt, strukturell, Regel-Ind., . . .)

Was ist Unifikation?

Unifikation:

- Verfahren, um zwei Terme identisch zu machen
- eventuell durch Ersetzen der schematischen Variablen durch Terme
- Anderes Beispiel: Pattern Matching

Beispiel: Unifikation der Terme ($?x, ?y, ?z$ Variablen, a Konstante)

$$f(?x, g(?x, ?x)) \qquad \text{und } f(h(?y), g(?z, h(a)))$$

Was ist Unifikation?

Unifikation:

- Verfahren, um zwei Terme identisch zu machen
- eventuell durch Ersetzen der schematischen Variablen durch Terme
- Anderes Beispiel: Pattern Matching

Beispiel: Unifikation der Terme ($?x, ?y, ?z$ Variablen, a Konstante)

$$f(?x, g(?x, ?x)) \qquad \text{und } f(h(?y), g(?z, h(a)))$$

1. Schritt:

Was ist Unifikation?

Unifikation:

- Verfahren, um zwei Terme identisch zu machen
- eventuell durch Ersetzen der schematischen Variablen durch Terme
- Anderes Beispiel: Pattern Matching

Beispiel: Unifikation der Terme ($?x, ?y, ?z$ Variablen, a Konstante)

$$f(h(?y), g(h(?y), h(?y))) \text{ und } f(h(?y), g(?z, h(a)))$$

1. Schritt: $?x = h(?y)$

Was ist Unifikation?

Unifikation:

- Verfahren, um zwei Terme identisch zu machen
- eventuell durch Ersetzen der schematischen Variablen durch Terme
- Anderes Beispiel: Pattern Matching

Beispiel: Unifikation der Terme ($?x, ?y, ?z$ Variablen, a Konstante)

$$f(h(?y), g(h(?y), h(?y))) \text{ und } f(h(?y), g(?z, h(a)))$$

1. Schritt: $?x = h(?y)$
2. Schritt:

Was ist Unifikation?

Unifikation:

- Verfahren, um zwei Terme identisch zu machen
- eventuell durch Ersetzen der schematischen Variablen durch Terme
- Anderes Beispiel: Pattern Matching

Beispiel: Unifikation der Terme ($?x, ?y, ?z$ Variablen, a Konstante)

$$f(h(?y), g(h(?y), h(?y))) \text{ und } f(h(?y), g(h(?y), h(a)))$$

1. Schritt: $?x = h(?y)$
2. Schritt: $?z = h(?y)$

Was ist Unifikation?

Unifikation:

- Verfahren, um zwei Terme identisch zu machen
- eventuell durch Ersetzen der schematischen Variablen durch Terme
- Anderes Beispiel: Pattern Matching

Beispiel: Unifikation der Terme ($?x, ?y, ?z$ Variablen, a Konstante)

$$f(h(?y), g(h(?y), h(?y))) \text{ und } f(h(?y), g(h(?y), h(a)))$$

1. Schritt: $?x = h(?y)$
2. Schritt: $?z = h(?y)$
3. Schritt:

Was ist Unifikation?

Unifikation:

- Verfahren, um zwei Terme identisch zu machen
- eventuell durch Ersetzen der schematischen Variablen durch Terme
- Anderes Beispiel: Pattern Matching

Beispiel: Unifikation der Terme ($?x, ?y, ?z$ Variablen, a Konstante)

$$f(h(a), g(h(a), h(a))) \qquad \text{und } f(h(a), g(h(a), h(a)))$$

1. Schritt: $?x = h(a)$
2. Schritt: $?z = h(a)$
3. Schritt: $?y = a$

Was ist Substitution?

Substitution:

Können einen Term durch einen anderen ersetzen, wenn beide gleich sind

Regel:

$$\frac{s = t \qquad P[s/x]}{P[t/x]}$$

$P[t/x]$: ersetze x in P durch t

Was ist Deduktion?

Deduktion:

- meist Inferenzregeln (aus Prämissen folgt Konklusion)
- für jedes logische Symbol zwei Regeln:

  Introduktion: wie erhalte ich diese Formel?

  Elimination: was kann ich aus dieser Formel folgern?

Was ist Deduktion?

Deduktion:

- meist Inferenzregeln (aus Prämissen folgt Konklusion)
- für jedes logische Symbol zwei Regeln:

  Introduktion: wie erhalte ich diese Formel?

  Elimination: was kann ich aus dieser Formel folgern?

Beispiel:

Introduktion von Konjunktion: $\dfrac{P \qquad Q}{P \wedge Q}$

Elimination von Konjunktion: $\dfrac{P \wedge Q \qquad \dfrac{P \qquad Q}{R}}{R}$

mehr in den Übungen!

Was ist Induktion?

Induktion:

- natürliche Induktion: zeige $P(0)$ und $P(n) \longrightarrow P(n+1)$
- strukturelle Induktion: Induktion über rekursive Datentypen
  Beispiel: Listen von natürlichen Zahlen $nat\ list = [] \mid nat\#nat\ list$
  ([] =leere Liste, $\#$ = Konkatenation)
  Induktion: zeige $P([])$ und $P(ns) \longrightarrow P(n\#ns)$
- wohlgeformte Induktion: Induktion über Relationen
  Beispiel: $<$ (auch: *starke Induktion*) $(\forall k < n.P(k)) \longrightarrow P(n)$

Was ist Induktion?

Induktion:

- natürliche Induktion: zeige $P(0)$ und $P(n) \longrightarrow P(n+1)$
- strukturelle Induktion: Induktion über rekursive Datentypen
  Beispiel: Listen von natürlichen Zahlen $nat\ list = [] \mid nat\#nat\ list$
  ([] =leere Liste, $\#$ = Konkatenation)
  Induktion: zeige $P([])$ und $P(ns) \longrightarrow P(n\#ns)$
- wohlgeformte Induktion: Induktion über Relationen
  Beispiel: $<$ (auch: *starke Induktion*) $(\forall k < n.P(k)) \longrightarrow P(n)$

Was ist Induktion?

Induktion:

- natürliche Induktion: zeige $P(0)$ und $P(n) \longrightarrow P(n+1)$
- strukturelle Induktion: Induktion über rekursive Datentypen
  Beispiel: Listen von natürlichen Zahlen $nat\ list = [] \mid nat\#nat\ list$
  ([] =leere Liste, $\#$ = Konkatenation)
  Induktion: zeige $P([])$ und $P(ns) \longrightarrow P(n\#ns)$
- wohlgeformte Induktion: Induktion über Relationen
  Beispiel: $<$ (auch: *starke Induktion*) $(\forall k < n.P(k)) \longrightarrow P(n)$

Was ist Induktion?

Induktion:

- natürliche Induktion: zeige $P(0)$ und $P(n) \longrightarrow P(n+1)$
- strukturelle Induktion: Induktion über rekursive Datentypen
  Beispiel: Listen von natürlichen Zahlen $nat\ list = [] \mid nat\#nat\ list$
  ([] =leere Liste, $\#$ = Konkatenation)
  Induktion: zeige $P([])$ und $P(ns) \longrightarrow P(n\#ns)$
- wohlgeformte Induktion: Induktion über Relationen
  Beispiel: $<$ (auch: *starke Induktion*) $(\forall k < n.P(k)) \longrightarrow P(n)$

Was ist Induktion?

Induktion:

- natürliche Induktion: zeige $P(0)$ und $P(n) \longrightarrow P(n+1)$
- strukturelle Induktion: Induktion über rekursive Datentypen
  Beispiel: Listen von natürlichen Zahlen $nat\ list = [] \mid nat\#nat\ list$
  ([] =leere Liste, $\#$ = Konkatenation)
  Induktion: zeige $P([])$ und $P(ns) \longrightarrow P(n\#ns)$
- wohlgeformte Induktion: Induktion über Relationen
  Beispiel: $<$ (auch: *starke Induktion*) $(\forall k < n.P(k)) \longrightarrow P(n)$

mehr in den Übungen!

Fazit

Theorembeweiser sind mächtiges Tool, aber kein 'goldener Hammer'!

- Kann Sicherheit bzgl. Aussagen beträchtlich erhöhen
- aber 'schnell mal etwas formalisieren und beweisen' unmöglich
- meistens werden Aussagen über Kernprobleme formalisiert und bewiesen

Fragen

Sind 'Papier und Bleistift' Beweise nicht einfacher?

Fragen

Sind 'Papier und Bleistift' Beweise nicht einfacher?

- Formalisierung in Theorembeweiser braucht viel Formalisierungsarbeit, auch für scheinbar 'triviale' Dinge

Fragen

Sind 'Papier und Bleistift' Beweise nicht einfacher?

- Formalisierung in Theorembeweiser braucht viel Formalisierungsarbeit, auch für scheinbar 'triviale' Dinge
- doch Beweise von Hand enthalten oftmals Fehler, vor allem für komplexe Strukturen

Fragen

Sind 'Papier und Bleistift' Beweise nicht einfacher?

- Formalisierung in Theorembeweiser braucht viel Formalisierungsarbeit, auch für scheinbar 'triviale' Dinge
- doch Beweise von Hand enthalten oftmals Fehler, vor allem für komplexe Strukturen
- Viel Aufwand, dafür garantierte Korrektheit!

Fragen

Wie kann ich sicher sein, dass meine Abstraktion das gewählte Problem beschreibt?

Fragen

Wie kann ich sicher sein, dass meine Abstraktion das gewählte Problem beschreibt?

- Im Allgemeinen: gar nicht!

Fragen

Wie kann ich sicher sein, dass meine Abstraktion das gewählte Problem beschreibt?

- Im Allgemeinen: gar nicht!
- Je genauer am konkreten Problem, desto größer die Sicherheit, aber 'Formalisierungslücke' bleibt