

Automatisierte Logik und Programmierung II

Sommersemester 2009



Christoph Kreitz

Theoretische Informatik, Raum 1.18, Telephon 3060

kreitz@cs.uni-potsdam.de

<http://www.cs.uni-potsdam.de/ti/lehre/alupII.htm>



1. Lehrziele
2. Rückblick & Ausblick
3. Organisatorisches

ZIEL DER VERANSTALTUNG

Computergestütztes logisches Schließen

ZIEL DER VERANSTALTUNG

Computergestütztes logisches Schließen

- **Mathematische Beweisführung**

- Aufdeckung und Korrektur von Fehlern (Beweisprüfung)
- Automatische Suche nach neuen Beweisen (Theorembeweisen)

Computergestütztes logisches Schließen

- **Mathematische Beweisführung**

- Aufdeckung und Korrektur von Fehlern (Beweisprüfung)
- Automatische Suche nach neuen Beweisen (Theorembeweisen)

- **Unterstützung für Entwurf zuverlässiger Software**

- Fehlersuche und Korrektheitsbeweise (Verifikation)
- Verbesserung der Performanz (Optimierung)
- Erzeugung aus Spezifikationen (Synthese)

ZIEL DER VERANSTALTUNG

Computergestütztes logisches Schließen

- **Mathematische Beweisführung**

- Aufdeckung und Korrektur von Fehlern (Beweisprüfung)
- Automatische Suche nach neuen Beweisen (Theorembeweisen)

- **Unterstützung für Entwurf zuverlässiger Software**

- Fehlersuche und Korrektheitsbeweise (Verifikation)
- Verbesserung der Performanz (Optimierung)
- Erzeugung aus Spezifikationen (Synthese)

- **Inferenzmaschine für KI-Systeme**

- Problemlöser und Planer für Roboter, ...

THEMEN DES ERSTEN SEMESTERS

Inferenzkalküle für Mathematik & Programmierung

Inferenzkalküle für Mathematik & Programmierung

- **Beweisen $\hat{=}$ Anwendung formaler Regeln**
 - Umgeht Mehrdeutigkeiten der natürlichen Sprache
 - Erlaubt schematische Lösung mathematischer Probleme

Inferenzkalküle für Mathematik & Programmierung

- **Beweisen $\hat{=}$ Anwendung formaler Regeln**
 - Umgeht Mehrdeutigkeiten der natürlichen Sprache
 - Erlaubt schematische Lösung mathematischer Probleme
- **Kernbestandteile:**
 - Formale Sprache (Syntax + Semantik)
 - Ableitungssystem (Axiome + Inferenzregeln)
 - Notwendige Eigenschaften: korrekt, vollständig, automatisierbar
 - Nützliche Eigenschaften: konstruktiv, ausdrucksstark, lesbar

Inferenzkalküle für Mathematik & Programmierung

- **Beweisen $\hat{=}$ Anwendung formaler Regeln**
 - Umgeht Mehrdeutigkeiten der natürlichen Sprache
 - Erlaubt schematische Lösung mathematischer Probleme
- **Kernbestandteile:**
 - Formale Sprache (Syntax + Semantik)
 - Ableitungssystem (Axiome + Inferenzregeln)
 - Notwendige Eigenschaften: korrekt, vollständig, automatisierbar
 - Nützliche Eigenschaften: konstruktiv, ausdrucksstark, lesbar
- **Vorgestellte Kalküle**
 - Prädikatenlogik (Logisches Schließen)
 - λ -Kalkül (Programmierung)
 - Einfache Typentheorie (Programmeigenschaften)
 - **Konstruktive Typentheorie (CTT)** (Uniformer Kalkül)

EIGENSCHAFTEN DER KONSTRUKTIVEN TYPENTHEORIE

- **Extrem ausdrucksstarkes Inferenzsystem**
 - Vereinheitlicht und erweitert Logik, λ -Kalkül & einfache Typentheorie
 - Direkte Darstellung der zentralen Konzepte (keine Simulation)
 - Formalisierung “natürlicher” Gesetze als Regeln

EIGENSCHAFTEN DER KONSTRUKTIVEN TYPENTHEORIE

- **Extrem ausdrucksstarkes Inferenzsystem**

- Vereinheitlicht und erweitert Logik, λ -Kalkül & einfache Typentheorie
- Direkte Darstellung der zentralen Konzepte (keine Simulation)
- Formalisierung “natürlicher” Gesetze als Regeln

- **Sehr umfangreicher Formalismus**

- Viele vordefinierte Basiskonstrukte, mehr als 150 Inferenzregeln
- Programmkonstruktion durch konstruktive Beweisführung möglich
- Abhängige Datentypen machen Wohlgeformtheit unentscheidbar
- Gestützt auf eigenständige konstruktive semantische Theorie

EIGENSCHAFTEN DER KONSTRUKTIVEN TYPENTHEORIE

- **Extrem ausdrucksstarkes Inferenzsystem**

- Vereinheitlicht und erweitert Logik, λ -Kalkül & einfache Typentheorie
- Direkte Darstellung der zentralen Konzepte (keine Simulation)
- Formalisierung “natürlicher” Gesetze als Regeln

- **Sehr umfangreicher Formalismus**

- Viele vordefinierte Basiskonstrukte, mehr als 150 Inferenzregeln
- Programmkonstruktion durch konstruktive Beweisführung möglich
- Abhängige Datentypen machen Wohlgeformtheit unentscheidbar
- Gestützt auf eigenständige konstruktive semantische Theorie

- **Probleme bei der praktischen Anwendung**

- Beweise erfordern viel Schreibarbeit → interaktive Beweissysteme
- Beweise sind unübersichtlich (komplexer Beweisbaum)
- Beweise oft schwer zu finden (viele Regeln und Parameter)
→ Automatisierung der Beweisführung

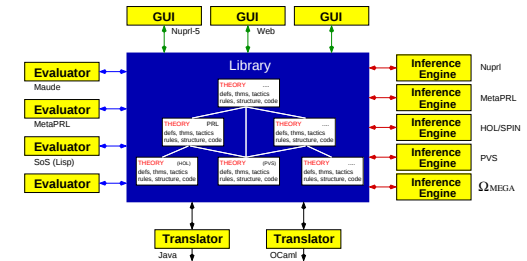
THEMEN DES ZWEITEN TEILS

Konstruktion und Einsatz von Beweissystemen

THEMEN DES ZWEITEN TEILS

Konstruktion und Einsatz von Beweissystemen

- Entwurf interaktiver **Beweisassistenten**
 - Das **NuPRL** Logical **P**rogramming **E**nvironment
 - Das **Isabelle** Beweissystem unter **ProofGeneral**



THEMEN DES ZWEITEN TEILS

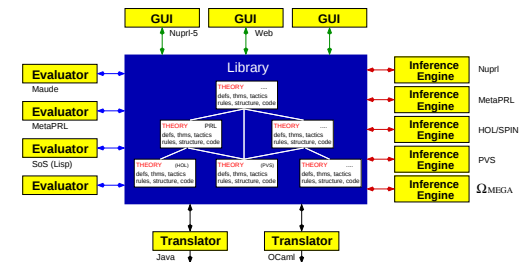
Konstruktion und Einsatz von Beweissystemen

- Entwurf interaktiver **Beweisassistenten**

- Das **NuPRL** Logical Programming Environment
- Das **Isabelle** Beweissystem unter **ProofGeneral**

- **Beweisautomatisierung**

- **Taktiken** und **Beweisplanung**: benutzerdefinierte Beweisstrategien
- **Entscheidungsprozeduren**: Lösung entscheidbarer Teilprobleme



THEMEN DES ZWEITEN TEILS

Konstruktion und Einsatz von Beweissystemen

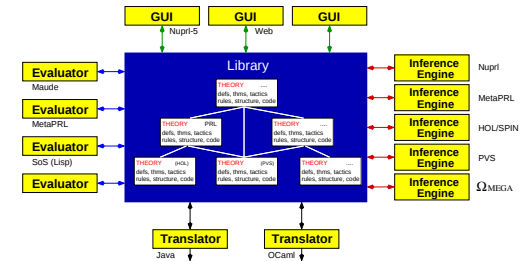
- Entwurf interaktiver **Beweisassistenten**

- Das **NuPRL** Logical Programming Environment
- Das **Isabelle** Beweissystem unter **ProofGeneral**

- **Beweisautomatisierung**

- **Taktiken** und **Beweisplanung**: benutzerdefinierte Beweisstrategien
- **Entscheidungsprozeduren**: Lösung entscheidbarer Teilprobleme

- **Wissensverwaltung und Benutzerinteraktion**

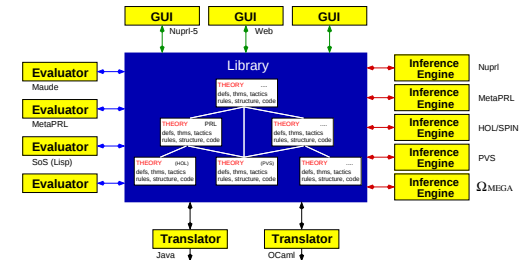


THEMEN DES ZWEITEN TEILS

Konstruktion und Einsatz von Beweissystemen

- **Entwurf interaktiver Beweisassistenten**

- Das **NuPRL** Logical Programming Environment
- Das **Isabelle** Beweissystem unter **ProofGeneral**



- **Beweisautomatisierung**

- **Taktiken** und **Beweisplanung**: benutzerdefinierte Beweisstrategien
- **Entscheidungsprozeduren**: Lösung entscheidbarer Teilprobleme

- **Wissensverwaltung und Benutzerinteraktion**

- **Anwendungen & Demonstrationen**

- Entwicklung formaler mathematischer Wissensbanken
- Wissensbasierte **Synthese von Programmen** aus Spezifikationen
- Korrektheitserhaltende **Optimierung von Algorithmen**

⋮

- **Zuordnung: theoretische/angewandte Informatik**

- **Zuordnung: theoretische/angewandte Informatik**
- **Veranstaltungsarten**
 - **Vorlesung**: Präsentation der zentralen Konzepte
 - **Übung**: Vertiefung und Anwendung theoretischer Aspekte
 - Optional: selbstgewähltes **praktisches Beweiserprojekt**

- **Zuordnung: theoretische/angewandte Informatik**
- **Veranstaltungsarten**
 - **Vorlesung**: Präsentation der zentralen Konzepte
 - **Übung**: Vertiefung und Anwendung theoretischer Aspekte
 - Optional: selbstgewähltes **praktisches Beweiserprojekt**
- **Veranstaltungstermine**
 - **Do 10:15–11:45** – Vorlesung
 - **Fr 12:30–14:00** – Vorlesung/Übung im Wechsel

- **Zuordnung:** **theoretische/angewandte Informatik**
- **Veranstaltungsarten**
 - **Vorlesung:** Präsentation der zentralen Konzepte
 - **Übung:** Vertiefung und Anwendung theoretischer Aspekte
 - Optional: selbstgewähltes **praktisches Beweiserprojekt**
- **Veranstaltungstermine**
 - **Do 10:15–11:45** – Vorlesung
 - **Fr 12:30–14:00** – Vorlesung/Übung im Wechsel
- **Lehrmaterialien:**
 - Vorlesungsskript von 1995, **Fachartikel** und **Manuals**

- **Zuordnung:** **theoretische/angewandte Informatik**
- **Veranstaltungsarten**
 - **Vorlesung:** Präsentation der zentralen Konzepte
 - **Übung:** Vertiefung und Anwendung theoretischer Aspekte
 - Optional: selbstgewähltes **praktisches Beweiserprojekt**
- **Veranstaltungstermine**
 - **Do 10:15–11:45** – Vorlesung
 - **Fr 12:30–14:00** – Vorlesung/Übung im Wechsel
- **Lehrmaterialien:**
 - Vorlesungsskript von 1995, **Fachartikel** und **Manuals**
- **Erfolgskriterien**
 - **Abschlußprüfung** (mündlich) **Aktive Teilnahme an Übungen wichtig**