Inferenzmethoden

Sommersemester 2015

Christoph Kreitz

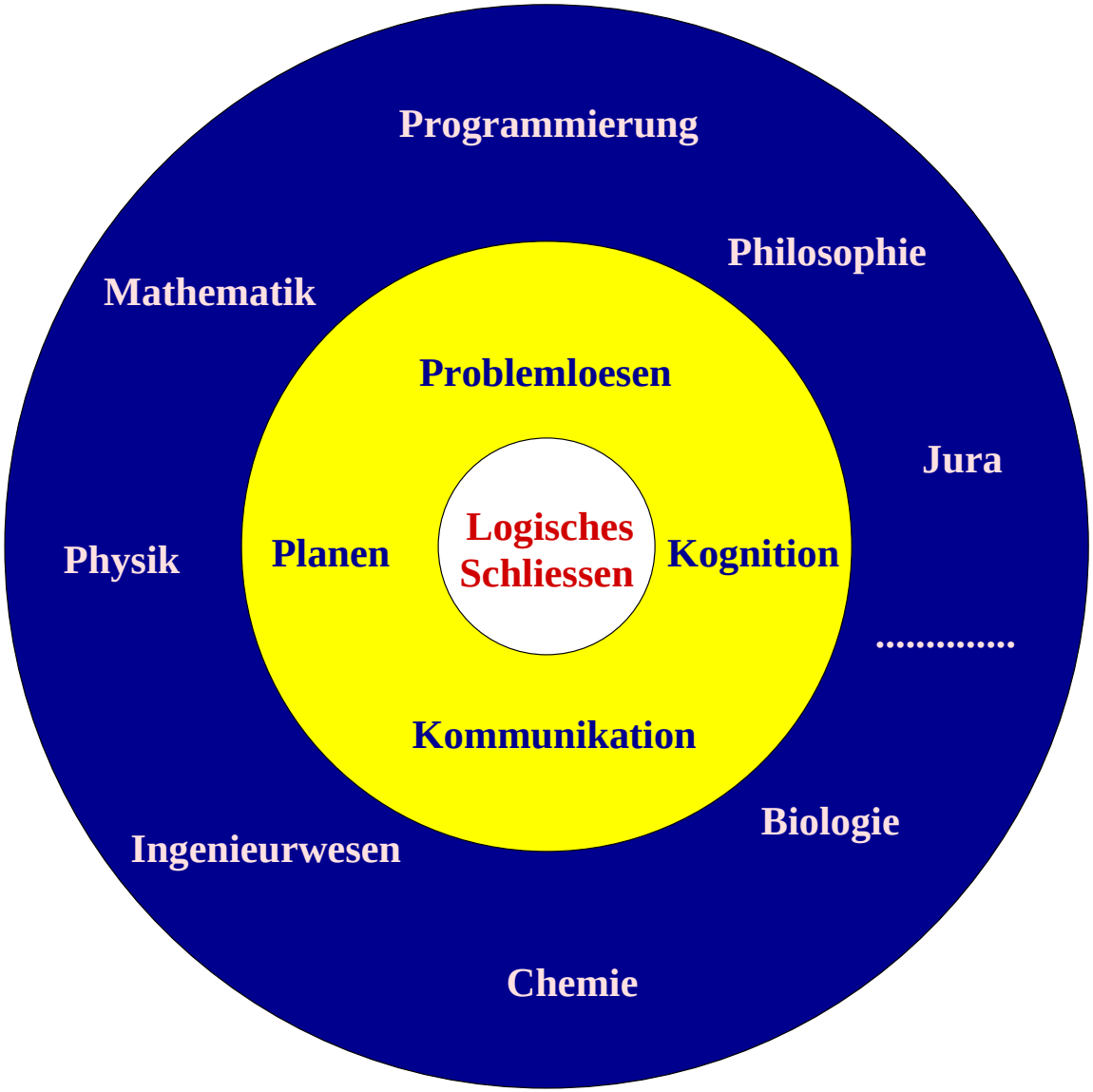
Theoretische Informatik, Raum 1.18, Telephon 3060

kreitz@cs.uni-potsdam.de

http://www.cs.uni-potsdam.de/ti

1. Ziele und Themen
2. Organisatorisches

Inferenzmethoden – wozu?



Logisches Schließen steht im Zentrum intelligenten Handelns

Logik präzisiert Schlussfolgerungen

- **Wissenschaft und Programmierung benötigt Vertrauen**
  - Warum sollten wir eine aufgestellte Behauptung akzeptieren?
    - gilt ein mathematisches Theorem wirklich?
    - ist es sicher, daß ein Programm wie vorgesehen funktioniert?
  - Welche Belege können wir für eine Behauptung geben?
    - was ist ein akzeptabler Beweis?
    - wie können wir Software verifizieren?
- **Es gibt viele Fehler in wissenschaftlichen Arbeiten**
  - Mathematische Analysen sind oft sehr kompliziert
  - Menschen machen Fehler bei der Ausarbeitung von Details
  - Fehler werden auch beim Reviewprozeß übersehen

    ca. 40–50% aller veröffentlichten Resultate sind falsch
- **Fast alle Softwareprodukte sind unzuverlässig**
  - Softwareprodukte und ihre Anforderungen sind extrem komplex
  - Auch sorgfältig getestete Programme enthalten größere Fehler

    ca 3-5 Fehler auf 1000 LOC “Hochqualitätscode” bei Kosten von $200/LOC

Was sind die Probleme?

- **Anfallende Aufgaben sind für Menschen zu komplex**
  - Menschen machen Fehler bei der Ausarbeitung der Lösung
  - Menschen machen Fehler beim Überprüfen komplexer Vorgänge
- **Fehlerquelle Oberflächlichkeit**

  Plausibilität wird mit Wahrheit verwechselt
  - Unzulässige Gedankensprünge, Analogien und Verallgemeinerungen
  - Übersehen möglicher Ausnahmesituationen Ariane 501
  - Plausible Schlußfolgerungen werden ohne Überprüfung akzeptiert
  - Behauptungen von “Autoritäten” werden ungeprüft übernommen
- **Fehlerquelle Überlastung**
  - Flüchtigkeitsfehler bei Behandlung von vielen Details Pentium I
  - Oft ist die Zeit zu knapp, um die “richtige” Lösung zu finden

Typische Fehler beim logischen Schliessen

- **Trugschlüsse**

  Tatsache: *Wenn es regnet, wird die Straße naß*

  Konsequenz: *Wenn die Straße nicht naß ist, hat es nicht geregnet* ✓

  Konsequenz: *Wenn die Straße naß ist, muß es geregnet haben* falsch!

Typische Fehler beim logischen Schliessen

- **Trugschlüsse**

  Tatsache: *Wenn es regnet, wird die Straße naß*

  Konsequenz: *Wenn die Straße nicht naß ist, hat es nicht geregnet* ✓

  Konsequenz: *Wenn die Straße naß ist, muß es geregnet haben* falsch!

- **Falsche Begründungen**

  Tatsache: *Alle Säugetiere sind behaart*

  Tatsache: *Alle Affen sind behaart*

  Konsequenz: *Also sind alle Affen Säugetiere*

Typische Fehler beim logischen Schliessen

- **Trugschlüsse**

  Tatsache: *Wenn es regnet, wird die Straße naß*

  Konsequenz: *Wenn die Straße nicht naß ist, hat es nicht geregnet* ✓

  Konsequenz: *Wenn die Straße naß ist, muß es geregnet haben* falsch!

- **Falsche Begründungen**

  Tatsache: *Alle Säugetiere sind behaart*

  Tatsache: *Alle ~~Affen~~ Teddybären sind behaart*

  Konsequenz: *Also sind alle ~~Affen~~ Teddybären Säugetiere* falsch!

  Ursprüngliche Konsequenz ist zufällig richtig aber nicht allgemeingültig

Typische Fehler beim logischen Schliessen

- **Trugschlüsse**

  Tatsache: *Wenn es regnet, wird die Straße naß*

  Konsequenz: *Wenn die Straße nicht naß ist, hat es nicht geregnet* ✓

  Konsequenz: *Wenn die Straße naß ist, muß es geregnet haben* falsch!

- **Falsche Begründungen**

  Tatsache: *Alle Säugetiere sind behaart*

  Tatsache: *Alle ~~Affen~~ Teddybären sind behaart*

  Konsequenz: *Also sind alle ~~Affen~~ Teddybären Säugetiere* falsch!

  Ursprüngliche Konsequenz ist zufällig richtig aber nicht allgemeingültig

- **Oberflächliche Betrachtung**

  Tatsache: 2^x *wächst viel schneller als jedes Polynom*

  Konsequenz: *Also ist* $x^2 < 2^x$ *für alle* x falsch für $x < 2$!

Wir brauchen Werkzeuge zur Unterstützung

Simuliere logisches Schliessen auf dem Computer

- **Es gibt großen Bedarf für intelligente Werkzeuge**
  - Automatische Steuerungsanlagen, Roboter, intelligente Agenten
  - Routineaufgaben in Verwaltung, Industrie und Forschung
  - Mathematische Beweise, Softwareentwicklung, -optimierung, -verifikation

Simuliere logisches Schliessen auf dem Computer

- **Es gibt großen Bedarf für intelligente Werkzeuge**
  - Automatische Steuerungsanlagen, Roboter, intelligente Agenten
  - Routineaufgaben in Verwaltung, Industrie und Forschung
  - Mathematische Beweise, Softwareentwicklung, -optimierung, -verifikation
- **Logisches Schließen ist oft sehr schematisch**
  - Mechanische Inferenztechniken können sinnvolle Unterstützung bieten

Simuliere logisches Schliessen auf dem Computer

- **Es gibt großen Bedarf für intelligente Werkzeuge**
  - Automatische Steuerungsanlagen, Roboter, intelligente Agenten
  - Routineaufgaben in Verwaltung, Industrie und Forschung
  - Mathematische Beweise, Softwareentwicklung, -optimierung, -verifikation
- **Logisches Schließen ist oft sehr schematisch**
  - Mechanische Inferenztechniken können sinnvolle Unterstützung bieten
- **Logisches Schließen kann präzisiert werden**
  - Begriffe werden durch symbolische Platzhalter ersetzt
  - Abstrakte Symbole beschreiben Zusammenhänge zwischen Aussagen
  - Formeln offenbaren Struktur von Argumenten
    z.B. ist $((M \Rightarrow H) \wedge (A \Rightarrow H)) \Rightarrow (A \Rightarrow M)$ sicher nicht allgemeingültig
  - Formale Logik: Universelle, akkurate Sprache
    + Regeln für schematische Lösung von Problemen

Simuliere logisches Schliessen auf dem Computer

- **Es gibt großen Bedarf für intelligente Werkzeuge**
  - Automatische Steuerungsanlagen, Roboter, intelligente Agenten
  - Routineaufgaben in Verwaltung, Industrie und Forschung
  - Mathematische Beweise, Softwareentwicklung, -optimierung, -verifikation
- **Logisches Schließen ist oft sehr schematisch**
  - Mechanische Inferenztechniken können sinnvolle Unterstützung bieten
- **Logisches Schließen kann präzisiert werden**
  - Begriffe werden durch symbolische Platzhalter ersetzt
  - Abstrakte Symbole beschreiben Zusammenhänge zwischen Aussagen
  - Formeln offenbaren Struktur von Argumenten
    z.B. ist $((M \Rightarrow H) \wedge (A \Rightarrow H)) \Rightarrow (A \Rightarrow M)$ sicher nicht allgemeingültig
  - Formale Logik: Universelle, akkurate Sprache
    + Regeln für schematische Lösung von Problemen
- **Computer sind ideal für diese Aufgabe**
  - Regelanwendung ist symbolische Manipulation von Text
  - Viele Probleme sind durch Austesten aller Möglichkeiten lösbar

Es gibt theoretische Grenzen

- **Arithmetik ist nicht voll axiomatisierbar**

Es gibt theoretische Grenzen

- **Arithmetik ist nicht voll axiomatisierbar**
- **Kein allgemeines Verfahren kann entscheiden**
  - ob eine gegebene logische Formel gültig ist
  - ob ein gegebenes Programm terminiert
  - ob ein gegebenes Programm korrekt ist
  - ob zwei Programme dieselbe Funktionalität haben

Es gibt theoretische Grenzen

- **Arithmetik ist nicht voll axiomatisierbar**
- **Kein allgemeines Verfahren kann entscheiden**
  - ob eine gegebene logische Formel gültig ist
  - ob ein gegebenes Programm terminiert
  - ob ein gegebenes Programm korrekt ist
  - ob zwei Programme dieselbe Funktionalität haben

Beweisverfahren müssen nach Beweisen suchen

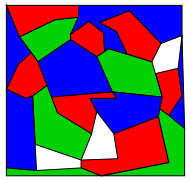
- Unendliche Suchbäume — keine Antwort im Mißerfolgsfall
- Suche erfordert Benutzersteuerung oder intelligente Strategien

Computergestütztes logisches Schliessen

- **Interaktive Beweisassistenten**
  - Benutzer konstruieren Beweise durch Anwendung von Regeln
  - Computer führt Regeln aus und zeigt ungelöste Teilprobleme
  - Mechanismus: Pattern Matching + Term Rewriting
- **Automatische Beweisprozeduren**
  - Taktiken: programmierte Anwendung von Inferenzregeln
  - Entscheidungsprozeduren for entscheidbare Teilprobleme
  - Beweissuchverfahren für eingeschränkte Anwendungsbereiche
    - Prädikatenlogik, Gleichheit, Induktion, Spezialanwendungen, ...
  - Beweisplaner, Model Checking, Computer Algebra, ...
- **Es gibt viele sinnvolle Anwendungsmöglichkeiten**
  - Mathematische Beweisführung
    - Beweisprüfung und automatisches Theorembeweisen
  - Unterstützung beim Entwurf zuverlässiger Software
    - Verifikation, kontrollierte Optimierung und Synthese
  - Problemlöser und Planer für KI-Systeme, Roboter, etc.

Erfolge der Vergangenheit machen Mut ...

1977: **Vier-Farben Problem**



– Spezialsoftware überprüft tausende kritischer Fälle

– Erneuter Beweis mit generischem Theorembeweiser Coq in 2005

1993: **Synthese von Scheduling Algorithmen**

– KIDS erzeugt korrekte Algorithmen in wenigen Stunden

– Erzeugter Lisp Code 2000 mal schneller als existierende ADA Software

1995: **Pentium Bug**

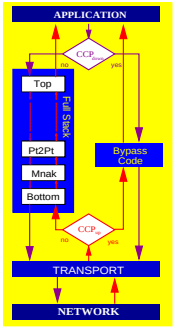
– Model Checking findet Fehler in Hardwaretabellen für Division

1996: **Robbins Hypothese**

The New York Times

– Theorembeweiser EQP findet (lesbaren) Beweis in 7 Tagen

1998: **Netzwerkverifikation, -optimierung, -entwurf**



– Verifikation findet subtilen Fehler in Kommunikationsprotokoll und ISO-Standards für Verschlüsselung

– Logische Optimierung steigert Performanz um Faktor 3–10

Seit 2012 werden für Hochsicherheitssoftware maschinenprüfbare Korrektheisbeweise verlangt

Der Schlüssel zum Erfolg ist Verdichtung

Entferne Redundanz aus Beweisen

- **Formale Logik und Semantik**
  - Repräsentation mathematischer Aussagen in präziser Sprache
  - Entfernt Mehrdeutigkeiten der natürlichen Sprache
- **Schematische Inferenzregeln für logische Konnektive**
  - Ersetzen mathematische Argumente auf Basis der formalen Semantik
  - Analytische Kalküle zerlegen Beweisaufgabe in kleinere Teilziele
- **Komprimierung strukturell gleichartiger Inferenzregeln**
  - Ersetzt Beweissuche auf Basis logischer Konnektive
- **Kompakte Beweisrepräsentation im Formelbaum**
  - Ersetzt ständige Erzeugung neuer Teilziele (Klauseln)
- **Zielorientierte Beweissuche**
  - Direkte Ansteuerung von beweisrelevanten Formelteilen und Instantiierung von Quantoren führt schneller zum Erfolg
- **Es gibt viele weitere Verdichtungsmöglichkeiten**

Themen dieser Veranstaltung

- **Logik und gültige Schlüsse**
  - Inferenzkalküle für die Prädikatenlogik erster Stufe
  - Refinement Logik und Tableauxbeweise
  - Maschinennahe Charakterisierung logischer Gültigkeit
- **Automatische Beweissuchverfahren**
  - Matrixmethoden für Aussagen- und Prädikatenlogik
  - Unifikationsalgorithmen
- **Implementierung effizienter Theorembeweiser**
  - Repräsentation von Formeln und Beweissuche
  - Effiziente Implementierung von Suchstrategien
  - Strategien zur Reduktion von Formeln und Suchraum
  - Methoden zur Messung der Leistungsfähigkeit von Beweisern
- **Jenseits von einfacher Prädikatenlogik**
  - Komplexere Logiken: konstruktive, modale und lineare Logik
  - Spezielle Theorien: Gleichheit, Arithmetik, Termauswertung
  - Wo geht es hin: Anwendungen und offene Fragen

Organisatorisches

- **Zuordnung: theoretische/angewandte Informatik**
- **Veranstaltungstermine**
  - Do 14:15–15:45 (0.02) – Vorlesung
  - Fr 12:15–13:45 (0.02) – Vorlesung/Übung im Wechsel
- **Empfehlenswerte Parallelveranstaltungen**
  - ATP Construction Project (M.Frank) für praktisch Interessierte
  - Geschichte der Logik (S.Böhne) liefert interessante Hintergünde
- **Lehrmaterialien:**
  - Buch “Deduktion” (online), Fachartikel, Folien der Veranstaltung
- **Erforderliche Vorkenntnisse:**
  - Grundkenntnisse in formaler Logik
- **Erfolgskriterien**
  - Abschlußprüfung (mündlich/schriftlich, je nach Teilnehmerzahl)
  - Aktive Teilnahme an Übungen sehr empfehlenswert