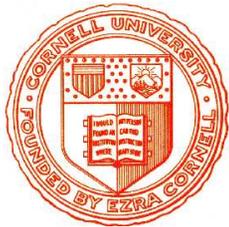


Automatisierte Logik und Programmierung

Einheit 23

Rückblick & Ausblick



1. Kalküle & Beweismethoden
2. Programmsynthese & Optimierung
3. Aktuelle Themen

RÜCKBLICK: KALKÜLE (ALuP I)

● Inferenzkalküle

- Formale Sprache zur Formulierung von mathematischen Problemen
- Regelsystem zum schematischen Beweisen mathematischer Aussagen
- Ausdrucksstarke Kalküle unterstützen formale Schlüsse über Programme
- Kalküle garantieren Korrektheit, sind aber keine Beweismethode

● Konstruktive Typentheorie (CTT)

- Vereinheitlicht und erweitert Logik, λ -Kalkül & einfache Typentheorie
- Umfangreicher Formalismus unterstützt viele Programmierkonstrukte
- Konstruktiv: unterstützt Extraktion von Programmen aus Beweisen
- Nichtkonstruktive Argumente können nicht geführt werden

Auch dort nicht, wo es auf die Konstruktion des Arguments nicht mehr ankommt

RÜCKBLICK: BEWEISSYSTEME

- **Interaktive Beweiseditoren**
 - Benutzergesteuerte Beweiskonstruktion
 - Unterstützen jeden Beweiskalkül, bieten aber keine Automatisierung
- **Taktisches Theorembeweisen**
 - Programmierte Anwendung von Inferenzregeln
 - Flexibel, sicher, mittlerer Automatisierungsgrad
- **Entscheidungsprozeduren**
 - Automatische Tests für entscheidbare Probleme
- **Theorembeweiser**
 - Vollständige Beweissuche in der Prädikatenlogik
- **Beweisassistenten**
 - Kombination verschiedener Techniken in einem System
 - Unterstützen Entwicklung mathematischer Theorien und Verwaltung großer Mengen mathematischen Wissens

- **Automatische Algorithmensynthese**

- Erzeugung korrekter ausführbarer Algorithmen aus Spezifikationen
- Programmiererfahrene Benutzer treffen zentrale Entwurfsentscheidungen
- System generiert Grundalgorithmus mit guter algorithmischer Struktur und garantiert Korrektheit des Entwurfs
Beispiele: Divide & Conquer, Globalsuche, Lokalsuche, Problemreduktionsgeneratoren
- Syntheseverfahren dokumentiert Entwurfsentscheidungen
(erleichtert spätere Modifikationen)

- **Optimierung und Datentypverfeinerung**

- Verbesserung des erzeugten Basisalgorithmus mit benutzergesteuerten logischen Optimierungstechniken
- Auswahl geeigneter Implementierungen für abstrakte Datentypen

Übertragung in konkrete Programmiersprache als letztes

PROGRAMMSYNTHESE: AUSBLICK

- **Hilfreich für die Praxis der Programmierung**
 - Aber Marktreife noch lange nicht erreicht
- **Zukunftsträchtiges Forschungsgebiet**
 - **Grundlagen:** Theoretische Analyse von Algorithmen
 - **Methoden:** Inferenz-, Synthese- und Optimierungsverfahren
 - **Korrektheit:** Einbettung in Beweisassistenten
- **Bedingungen an konkrete Systeme**
 - Interne Verarbeitung **formal korrekt**
 - **Externe Präsentation** möglichst wenig formal
 - **Graphische Unterstützung** für Kontrolle einer Synthese
 - **Große Wissensbanken** mit effizienter Verwaltung
- **Voraussetzung an Entwickler von Systemen**
 - Theoretische Grundlagen **und** praktische Programmierarbeiten
 - **Formales Denken**, Kenntnis logischer Kalküle, **Abstraktionsvermögen**
 - **Kreativität**, Experimentierfreudigkeit, **Ausdauer**, Frustrationstoleranz

AKTUELLE FRAGESTELLUNGEN

- **Theoretische Arbeiten**

- Integration klassischer Schlußfolgerungen in den konstruktiven Kalkül
- Neue Typkonstrukte für die Programmierung

- **Beweisverfahren**

- Neue Entscheidungsprozeduren und Theorembeweisertechniken
- Kooperierende & verteilte Beweiser
- Mechanismen zum Aufbau und Strukturierung einer Wissensbank
- Anwendungsspezifische Beweis- und Planungstechniken

- **Synthese- und Optimierungsverfahren**

- Integration von Synthese- und Optimierungsverfahren in Beweissysteme
- Entwurf/Verfeinerung neuer Algorithmentheorien

- **Konkrete Anwendungen der Verfahren**

Themen für Beleg-/Forschungs-/Diplomarbeiten

- **Nichtkonstruktive Beweisführung in der CTT**

- Ergänze “Extraktterme” für klassische Argumente
- Ausgangspunkt: $\lambda\mu\nu$ -Kalkül von Pym & Ritter
- Erweitere CTT-Sequenzen auf mehrere (benamte) Konklusionen
- Modifiziere $\lambda\mu\nu$ -Kalkül in Top-Down Sequenzenkalkül
- Formuliere Urteils-Semantik, Regeln, Korrektheitsbeweise, Typecheck, ...
- Ziel: klassisches Schließen ist erlaubt, aber im Extrakt identifizierbar
 - Extraktion von Algorithmen muß klassische Teilterme eliminieren
- Vorläufig Simulation in Nuprl durch (neuen) Komma-Typ und Taktiken
- Langfristig: Erweiterung von Nuprl auf Basis theoretischer Erkenntnisse

- **Neue Typkonstrukte für die Programmierung**

- Erweiterungen für verteilte Systeme
- Komplexitätsbehandlung und Sicherheit

THEMENKOMPLEX BEWEISTECHNIKEN

- **Taktische Beweisführung**
 - Aufbau einer vollständigen Beweistaktik für Prädikatenlogik
 - Integration von Standardtechniken wie Arithmetik, Induktion, ...
- **Rewrite-Techniken und Vorwärtsinferenz**
 - Umschreiben von Teiltermen auf der Basis von Lemmata
 - Kontrolle durch Vorgabe eines syntaktischen Beweisziels
- **Erweiterung der Nuprl Beweisverfahren**
 - Entscheidungsprozeduren (Aussagenlogik, datentypspezifisches, ...)
 - Theorembeweiser: jenseits von reiner Prädikatenlogik
 - Induktions- und Beweisplanungstechniken, Computeralgebra, ...
- **Wissensaufbau und -Organisation**
 - Erzeugung verifiziertes Wissen zu verschiedenen Anwendungsbereichen
 - Automatische Auswahl und Anwendung relevanter Lemmata
- **Benutzerinterface & Dokumentation**
 - Fortführung/Ergänzung einer laufenden Diplomarbeit

- **Einbettung von Syntheseverfahren in Nuprl**

- Implementierung eines Rahmenformalismus für Programmentwicklung
- Formale Korrektheitsbeweise für konkrete Algorithmentheorien
- Implementierung von Taktiken zur Bestimmung konkreter Parameter
- Extraktion lauffähiger Algorithmen
- Ankoppelung konkreter Programmiersprachen

- **Einbettung von Optimierungsverfahren in Nuprl**

- Taktiken zur Simplifikation, Endlichen Differenzierung, Fallanalyse, ...
- Automatische Erzeugung von Korrektheitsbeweisen für das Resultat

- **Weiterentwicklung neuer Algorithmentheorien**

- Verfeinerung der Technik für Problemreduktionsgeneratoren
- Entwurf verteilter / probabilistischer Algorithmen

THEMENKOMPLEX: ANWENDUNGEN

- **Mathematik**

- Automatisierung “trivialer” Beweise in der Kategorientheorie
- Graphentheoretische Konzepte und Algorithmen

⋮

- **Informationssicherheit**

- Unterstützung eines Forschungsprojektes an der RWTH Aachen
 - Verifikation eines Regelsystems zur Prüfung von Sicherheitsbedingungen
 - Verifikation eines Regelsystems zur Transformation in sicheren Code
 - Verifikation der Vollständigkeit eines Security-Transformers

- **Verteilte Systeme**

- Verifikation globaler Eigenschaften lokaler Kommunikationsprotokolle

⋮