Automatisierte Logik und Programmierung

Lektion 20

Rückblick & Ausblick

1. Kalküle & Beweismethoden
2. Programmsynthese & Optimierung
3. Wo geht es hin

Rückblick: Kalküle & Beweissysteme

- **Inferenzkalküle**
  - Formale Sprache zur Formulierung von mathematischer Problemen
  - Regelsystem zum schematischen Beweisen mathematischer Aussagen
  - Ausdrucksstarke Kalküle unterstützen formale Schlüsse über Programme
  - Kalküle garantieren Korrektheit, sind aber keine Beweismethode
- **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 for entscheidbare Probleme
  - Theorembeweiser: vollständige Beweissuche in Prädikatenlogik
  - Beweisplaner: Suche nach Beweis“skizzen” auf Meta-Ebene

  Beweisassistenten kombinieren verschiedene Techniken in einem System

Rückblick: Programmsynthese & Optimierung

- **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
  - 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 Forschungsthemen

- **Aufbau und Strukturierung einer Wissensbank**
  - Verifiziertes Wissen zu verschiedenen Anwendungsbereichen
  - Automatische Auswahl und Anwendung relevanter Lemmata
- **Beweisverfahren**
  - Entscheidungsprozeduren (Aussagenlogik, datentypspezifisches, ...)
  - Theorembeweiser: jenseits von reiner Prädikatenlogik
  - Induktions- und Beweisplanungstechniken, Computeralgebra, ...
  - Entwurf anwendungsspezifischer Verfahren
  - Kooperierende & verteilte Beweiser
- **Synthese- und Optimierungsverfahren**
  - Synthese auf Basis formaler Theoreme über Algorithmentheorien
  - Optimierung als verifizierter Vorwärtsinferenz
  - Einbettung konkreter Programmiersprachen
- **Benutzerinterface & Dokumentation**

$\Downarrow$

Forschungsprojekte & Diplomarbeiten