

Maschinelles Beweisen mit PVS

Vorlesung mit Übung
dazu gleichnamiges Praktikum

WS 2005/2006

F. von Henke, H. Pfeifer

Maschinelles Beweisen

Maschinelles Beweisen generell:

Logische Ableitung oder logisches Schließen mit Unterstützung der Maschine, d.h. des Rechners

- nach den Regeln einer formalen Logik (z.B. Prädikatenlogik) – die Logik definiert, was ein 'Beweis' ist.
- Maschinelle Behandlung ("Mechanisierung") des Beweisens soll dazu führen, dass der Vorgang des Beweisens und das Resultat (der Beweis) nachvollziehbar, überprüfbar und wiederholbar werden.

Historisches

- Idee des Theorembeweisens mit maschineller Unterstützung existiert seit Beginn der Computer-Entwicklung in den 40er Jahren : A. Turing
- Wichtiges Gebiet in den Anfängen des Gebiets der Künstlichen Intelligenz (KI):
J. McCarthy (1962) - *proof checking*
A. Robinson (ca 1969) - Resolutionskalkül
- intensive Entwicklung von Theorembeweisern in den 70er und 80er Jahren
Fokus des Interesses hauptsächlich auf “automatischem Beweisen”: der Rechner soll einen Beweis (zu einem mathematischem Theorem, . . .) selbständig finden
- zunächst hauptsächlich: Beweisen von mathematischen Theoremen
- Querverbindung zur (Programm-) *Verifikation*: z.B. Beweis von Verifikationsbedingungen, die bei Anwendung des Hoare-Kalküls erzeugt werden.

Historisches (2)

- parallele Entwicklungen:
 - Beweiser für Prädikatenlogik erster Stufe, überwiegend mit Resolution
 - Entwicklung von *Entscheidungsprozeduren* für spezielle Theorien:
 - Aussagenlogik
 - lineare Arithmetik
 - einfache Datenstrukturen (Arrays, Listen, . . .)
 - Termersetzungssysteme: Beweisen in Gleichungslogik
- heute: Mathematiker sind i.a. nicht besonders interessiert an Theorembeweisern als Werkzeug

aber: Kombinationen von Theorembeweisern und Systemen für *symbolisches Rechnen* (z.B. Mathematika, Maple) werden entwickelt

auch: Bestrebungen, mathematische Inhalte (einschließlich zugehöriger Ableitungen) durchgängig maschinell zu formalisieren und für Präsentation aufzubereiten (etwa online im Internet).

Historisches (3)

Hauptinteresse heute bei

- Modellierung und formaler Analyse kritischer Systeme: Sicherheit, Zuverlässigkeit, Vertrauenswürdigkeit, . . . von Systemen und System-Komponenten

Beweiser als notwendige Inferenz-Komponente beim praktischen Einsatz formaler Methoden
- Beweiser-Module als Hilfskomponenten zur Unterstützung von Inferenzen (“eingebettete Intelligenz”)
- “proof-carrying Code”
- typischerweise Einsatz von Kombinationen von Methoden, weniger Ableitung in einem ‘reinen’ Kalkül

Grade der Automatisierung

- **Beweisprüfung** (*proof checking*): Maschine überprüft einen vorgelegten Beweis daraufhin, ob es sich tatsächlich um einen Beweis handelt.
- **Automatisches Beweisen**: Maschine versucht, einen Beweis “automatisch”, d.h. ohne Mitwirkung eines menschlichen Benutzers, zu konstruieren.
- **Interaktives Beweisen**: Maschine agiert als “Beweisassistent” eines Benutzers
 - Routine-Aktivitäten werden so weit wie möglich von der Maschine übernommen
 - Benutzer hat (manuelle) Kontrolle auf der obersten Ebene der Beweis-Konstruktion
- **Taktisches Beweisen**: Verwendung von “Taktiken” oder “Strategien” – programmierte Beweiskonstruktion bzw. Beweissuche

Grade der Automatisierung (2)

Voll-automatisches Beweisen ist inhärent schwierig bzw. prinzipiell unmöglich, wegen

- Unentscheidbarkeit der Prädikatenlogik erster Stufe, bzw.
- Komplexität der Entscheidungsprobleme für einfachere Logiken (Aussagenlogik u.a.)

Je einfacher die Logik ist, desto größer ist die Chance des vollautomatischen Beweisens

Beispiel: gewisse Klassen von aussagenlogischen Problemen lassen sich mit neueren Ansätzen wie *binary decision diagrams* oder SAT-Solvern effizient behandeln.

Logik vs. Spezifikationsprache

Reine Logik (ganz gleich welcher Art) ist zu ausdrucksarm als Grundlage für maschinelles Beweisen.

- Mathematische Praxis: Beweise werden im Rahmen mathematischer *Theorien* geführt.

Beispiel: Gruppentheorie, charakterisiert durch die Gruppenaxiome

- Theorien brauchen einen syntaktischen Rahmen
 - Strukturierung
 - Komposition von Theorien
 - Lesbarkeit
- *Typisierung* genauso hilfreich wie in der Programmierung
- Unterstützung des Anlegens und Verwaltens von Bibliotheken (*libraries*) von Theorien, Theoremen, . . .

↪ Verwendung einer *Spezifikationsprache*

PVS

PVS – “Prototype Verification System”

- entwickelt im CSL von SRI International (Kalifornien) seit ca. 1990
- verfügbar seit 1992
- weltweit von den verschiedensten Gruppen eingesetzt, u.a. NASA (Formal Methods Group)
(vgl. nicht mehr ganz aktuelle Webseite bei SRI)
- Integriertes System für Spezifikation und Verifikation (Beweisen)
 - Logik höherer Ordnung als Grundlage
 - sehr ausdrucks mächtige Spezifikations sprache
 - weitreichendes Typsystem
 - interaktiver Beweiser
 - integrierte Entscheidungsprozeduren

Warum PVS?

Es gibt eine Reihe ähnlicher Systeme, u.a.

- Isabelle
- HOL (“Higher-order Logic”) - mehrere Varianten
- ACL2 (“A Computational Logic 2”)
- Coq - basiert auf Typentheorie

PVS wird benutzt, weil

- die Fähigkeiten des Systems angemessen sind;
- Alternativen nicht sichtbar “besser” sind;
- das System weltweit anerkannt ist;
- in Ulm langjährige und umfangreiche Erfahrungen mit dem System in gewonnen wurden.

“Typische” Einsatzgebiete von PVS

in Ulm:

- Modellierung und Analyse Fehlertoleranter verteilter Algorithmen insbesondere im Zusammenhang mit der “Time-triggered Architecture”
- Formalisierung der Semantik von Programmiersprachen, zusammen mit Basis-Theorien (Fixpunkttheorie u.a.)
- Compiler-Verifikation
- Programmtransformation
- Hardware-Modellierung und -Verifikation

weltweit: es gibt praktisch kein Gebiet der Anwendung formaler Methoden, in dem nicht auch PVS eingesetzt wurde oder wird.

Themengebiete

1. Logische Grundlagen:

- Aussagenlogik
- Prädikatenlogik 1. Stufe
- Prädikatenlogik höherer Stufe (*Higher-order Logic*)
(getypter) Lambda-Kalkül
- Sequenzenkalkül

2. Die Spezifikationssprache von PVS

- Struktur von Theorien
- Deklarationen
- Typsystem
- Abstrakte Datentypen
- Parametrisierung
- Rekursive und induktive Definitionen
- “prelude”: die Bibliothek von vordefinierten Theorien

Themengebiete (2)

3. Beweismethoden

- Interaktive Beweisentwicklung im Sequenzenkalkül
- Entscheidungsprozeduren
- Termersetzung (*rewrite rules*)
- Taktiken
- Beweis durch Induktion
- Modell-Überprüfung (*model checking*)

4. PVS-spezifische Techniken

- der Spezifikation
- des Beweisens