Analyzing Java in Isabelle/HOL

— Formalization, Type Safety and Hoare Logic —

Promotionsvortrag und -prüfung

9. Februar 2001

Dipl.-Inform. David von Oheimb, TU München

Vorsitzender: Univ.-Prof. Dr. Jürgen Eickel, TU München

1. Prüfer: Univ.-Prof. Tobias Nipkow, Ph.D., TU München

2. Prüfer: Univ.-Prof. Dr. Arnd Poetzsch-Heffter, FernUniversität Hagen

Motivation

IT-Einsatz in kritischen Bereichen

Zentrales Problem der Informatik:

Vermeidung von Software-Fehlern

Sicherste Lösung:

Maschinelle Verifikation

Grundlagen: Logik, Semantik

Technologie: interaktive Beweissysteme

Aktuell: Anwendung, Methodik

Beitrag: exemplarische Grundlagenarbeit

These:

Anwendungsreife erreicht

Überblick

Gesamtziel: Anwendung der Theorie und Technologie

zur Behandlung der Sprache JAVA mit dem Beweissystem Isabelle HOL

Spracherweiterungen | Programmverifikation | Compilerverifikation

Typsicherheit | Hoare-Logik | JVM-Formalisierung

S p r a c h - F o r m a l i s i e r u n g

Isabelle/HOL | Programmiersprachen-Semantik | Java

1. Sprach-Formalisierung

Zweck: Modell der Programmiersprache in der Logik (hier: HOL)

Ziele: Einfachheit, Treue zur Sprachspezifikation, Anwendbarkeit

Herausforderungen: Interaktion der Konzepte, Sprachumfang

Standardumfang: Klassen, Interfaces, Objekte, Methoden, Exceptions

Vorarbeiten: Nipkow: IMP, Drossopoulou et al. : Java_S

Entscheidungen: Operationelle Auswertungssemantik, Exceptions

$$prog \vdash (xcpt, state)\ -term \succ val \rightarrow\ (xcpt', state')$$

Erfahrungen

- Fehlervermeidung durch maschinelle Überprüfung
- Komplexitätsmanagement dringend erforderlich
  Techniken: Strukturierung, Abstraktion, Redundanzverminderung
- Erweiterungen gut möglich (eigene, Büchi)

Ergebnisse

- Grundlage für nachfolgende Arbeiten
- minimale Klärungen und Verallgemeinerungen der Spezifikation
- Entwicklungsstimulation für Isabelle/HOL

Künftige Arbeiten: weitere Validierung, Erweiterungen ($\leadsto$ Java Card)

Anwendungen: Sprachdesign-Studien, Compilerverifikation, Lehre (μJava)

2. Typsicherheits-Beweis

Begriff der Typsicherheit: Operationen nur auf “passenden” Argumenten

Bedeutung: wichtiges Designziel, Bestandteil des Sicherheitskonzepts

Hinreichend: “Well-typed programs do not go wrong”

Beweis
- Entwicklung von Hilfskonzepten wie Konformität
- ~ 300 Lemmata, Hauptbeweis per Regelinduktion

Erfahrungen: Auswertungssemantik ist große Vereinfachung

Ergebnisse:
- Java$^{\ell ight}$ ist tatsächlich typsicher
- Formalisierung für Sprachanalyse gut geeignet
- Mechanischer Theorembeweiser erzwingt Sauberkeit, nimmt Detailarbeit ab und unterstützt bei Änderungen

3. Hoare-Logik

Zweck: Basis für Korrektheitsbeweis von Programmen

$$prog, triples \vdash \{assn\}\ term\ \{assn\}$$

Ziele: Korrektheit relativ zur Op. Semantik, Anwendbarkeit, Vollständigkeit

Herausforderungen: Seiteneffekte, dynamische Bindung, Exceptions, Vollständigkeitsbeweis bei verschränkter Rekursion, Klasseninitialisierung

Vorarbeiten: Kleymann: IMP-Erweiterung in LEGO

Vorstudie: IMP + Prozeduren mit Parametern, verschränkte Rekursion

Zusicherungen: Flache Einbettung, Hilfsvariablen, Resultatbehandlung

Erweiterungen: Konsequenzregel, Exceptions, Zustandsabhängigkeit

Maschinengestützte Beweise

- Korrektheit per Regelinduktion (Invarianten, Annahmen)
- Vollständigkeit (relativ, $\rightarrow$ Cook) mit Most-General-Formula-Ansatz
  Strukturelle Induktion (Hilfsind. für Methoden, Klasseninitialisierungen)

Erfahrungen:
- Typsicherheit auch für Hoare-Logik notwendig
- (Beispiel $\Rightarrow$) Programmverifikation inhärent aufwendig, Maschinenunterstützung wesentlich

Ergebnisse:
- Logik für großen Sprachumfang, korrekt und vollständig
- Klassische Hoare-Logik auf Objektorientierung übertragbar

Künftige Arbeiten:
- Werkzeugunterstützung (EU-Projekt VerifiCard)
- Anbindung an Formalisierung der OCL

Zusammenfassung

- Adäquate Formalisierung einer realistischen Sprache
- Überprüfung des Java-Designs durch maschinellen Typsicherheitsbeweis
- Umfassendste Axiomatische Semantik für objektorientierte Sprache
- Erster Vollständigkeitsbeweis für eine OO-Hoare-Logik

Eine weitgehende und verlässliche Analyse realer Programmiersprachen ist mit Hilfe mechanischer Theorembeweiser inzwischen gut möglich.