Formale Spezifikation und Verifikation SoSe 17

Übung 01

Philipp Wendler

10. Mai 2017

Organisatorisches

- Bitte im UniWorX in Veranstaltung eintragen.
- Informationen zur Veranstaltung auf der Webseite (im UniWorX verlinkt).
- Klausur:
  - 1. Termin voraussichtlich 31. Juli
  - 2. Termin im Oktober vor den Vorlesungen
  - Anmeldung über UniWorX nötig
- Übungsblätter:
  - Am Anfang wöchentlich, später größeres Projekt
  - Ausgabe über UniWorX jeweils Ende der Woche
  - keine Abgabe, keine Bewertung
  - Besprechung in der Übung

Modelle

Welche Modelle für Software kennen Sie?
Welche eigenen sich für Verifikation?

Modelle

- Petri-Netz
- Prozess-Algebren
- Automat, Transitionssystem
- Klassendiagramm, Sequenzdiagramm
- Modellierungssprachen (für Model Driven Software Design)
- Source-Code, Binär-Code, Zwischensprachen
- Compiler-interne Repräsentation (Control Flow Graph, Abstract Syntax Tree)

Spezifikation

Wie kann eine Spezfikation für ein System angegeben werden?

Spezifikation

- Direkt im System:
  - `assert` Statements
  - Gekennzeichnete Fehlerbereiche, die nicht erreicht werden dürfen
- Temporallogische Formeln
- Automaten
- Informal, z.B.
  - Keine `null`-pointer dereferences
  - Keine Deadlocks

Verifikation

"Beware of bugs in the above code; I have only proved it correct, not tried it."

Donald E. Knuth (`https://cs.stanford.edu/~uno/faq.html`)

Software-Qualitätssicherung

Welche Methoden gibt es, um Fehler in Software zu finden?

Software-Qualitätssicherung

- Testen
- Typchecks
- Code Reviews
- Statische Analysen (Lint, FindBugs, Google Error-Prone)
- Automatische Verifikation (Model Checking)
- Interaktive Verifikation (mit Theorem Provern)
- Manuelle Verifikation