Formale Programmverifikation

Referentin: Kirsten Hradek

Formale Programmverifikation

Formale Programmverifikation

- Einführung/Motivation

Formale Programmverifikation

- Einführung/Motivation
- Softwareverifikation am Beispiel des Hoare-Kalküls

Formale Programmverifikation

- Einführung/Motivation
- Softwareverifikation am Beispiel des Hoare-Kalküls
- Überblick über verschiedene Verifikationsmethoden

Formale Programmverifikation

- Einführung/Motivation
- Softwareverifikation am Beispiel des Hoare-Kalküls
- Überblick über verschiedene Verifikationsmethoden
- Rechnerunterstützte Verifikationsmethoden

Formale Programmverifikation

- Einführung/Motivation
- Softwareverifikation am Beispiel des Hoare-Kalküls
- Überblick über verschiedene Verifikationsmethoden
- Rechnerunterstützte Verifikationsmethoden
- Möglichkeiten und Grenzen der SV

Formale Programmverifikation

- Einführung/Motivation
- Softwareverifikation am Beispiel des Hoare-Kalküls
- Überblick über verschiedene Verifikationsmethoden
- Rechnerunterstützte Verifikationsmethoden
- Möglichkeiten und Grenzen der SV

Das Grundproblem:

Das Grundproblem:

Qualitätskrise

Das Grundproblem: Qualitätskrise

Software wird zur Benutzung freigegeben, nicht wenn sie nachweislich korrekt ist, sondern wenn die Häufigkeit, mit der neue Fehler entdeckt werden, auf ein für die Geschäftsleitung akzeptables Niveau gesunken ist. DAVID L. PARNAS

Das Grundproblem: Qualitätskrise

Software wird zur Benutzung freigegeben, nicht wenn sie nachweislich korrekt ist, sondern wenn die Häufigkeit, mit der neue Fehler entdeckt werden, auf ein für die Geschäftsleitung akzeptables Niveau gesunken ist. DAVID L. PARNAS

Jedes sechste DV-Projekt wurde ohne jegliches Ergebnis abgebrochen, alle Projekte überzogen die Zeit- und Kostenrahmen um 100-200% und auf 100 ausgelieferte Programmzeilen kommen im Durchschnitt drei Fehler. TOM DE MARCOS

Was bedeutet überhaupt Programmverifikation?

Was bedeutet überhaupt Programmverifikation?

Definition:
Programmverifikation ist ein systematischer Ansatz zum Nachweis der Fehlerfreiheit von Programmen. Dabei wird bewiesen, daß ein vorgegebenes Programm bestimmte wünschenswerte Eigenschaften besitzt.

(Apt/Olderog, 1998, 1)

Welche Fehler können auftreten?

Welche Fehler können auftreten?

- Syntaktische Fehler

Welche Fehler können auftreten?

- Syntaktische Fehler
- Semantische Fehler

Programmverifikation zur Qualitätssicherung?

Programmverifikation zur Qualitätssicherung?

- Einsetzbarkeit in der Praxis?

Programmverifikation zur Qualitätssicherung?

- Einsetzbarkeit in der Praxis?
- Effizienz?

Programmverifikation zur Qualitätssicherung?

- Einsetzbarkeit in der Praxis?
- Effizienz?
- Aufwand?

Programmverifikation zur Qualitätssicherung?

- Einsetzbarkeit in der Praxis?
- Effizienz?
- Aufwand?
- Kosten?

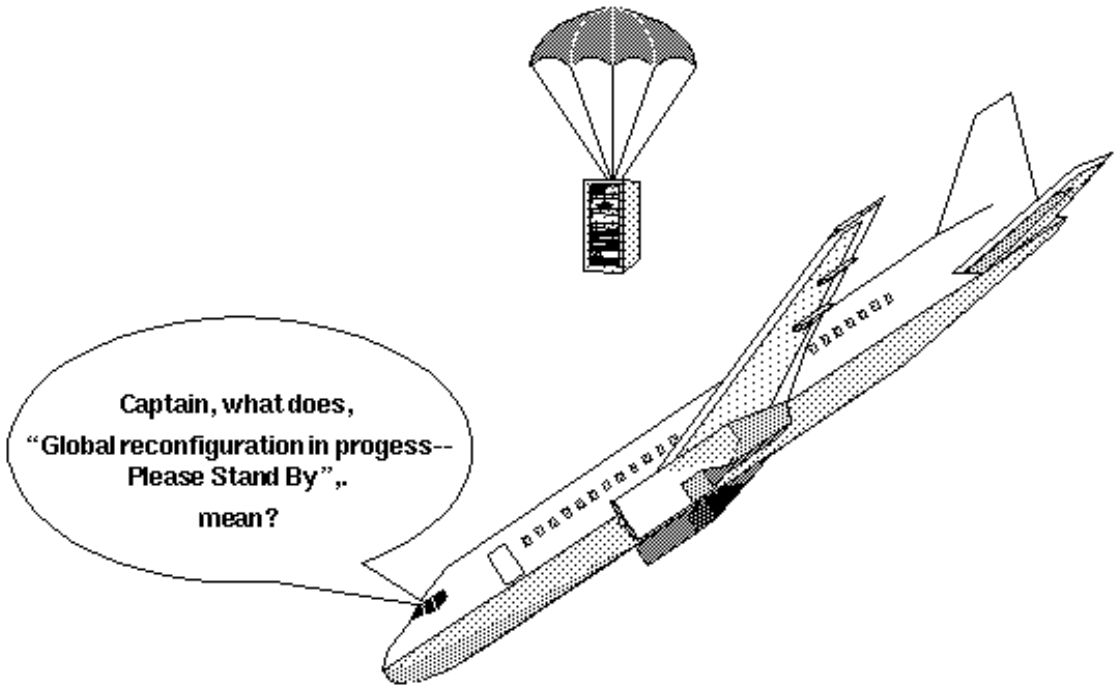
Programmverifikation zur Qualitätssicherung?

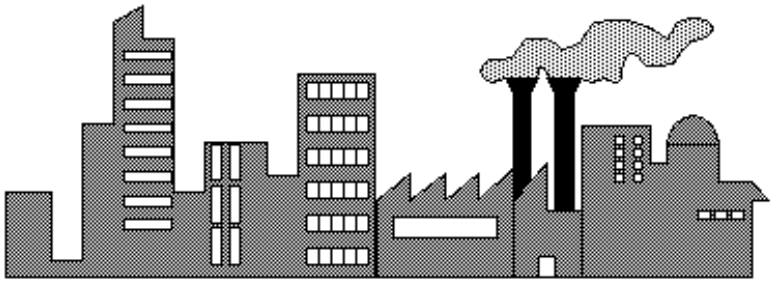
- Einsetzbarkeit in der Praxis?
- Effizienz?
- Aufwand?
- Kosten?
- Handhabbarkeit?

Programmverifikation zur Qualitätssicherung?

- Einsetzbarkeit in der Praxis?
- Effizienz?
- Aufwand?
- Kosten?
- Handhabbarkeit?
- Vorteile gegenüber anderen Methoden?

Programmverifikation?





Formale Programmverifikation

- Einführung/Motivation
- Softwareverifikation am Beispiel des Hoare-Kalküls
- Überblick über verschiedene Verifikationsmethoden
- Rechnerunterstützte Verifikationsmethoden
- Möglichkeiten und Grenzen der SV

Spezifikation

Spezifikation

- Präzise Darstellung und Beschreibung der Eigenschaften, der Verhaltensweisen, des Zusammenwirkens und des Aufbaus von (bestehenden oder zu entwickelnden) Modellen, Programmen oder Systemen

Spezifikation

- Präzise Darstellung und Beschreibung der Eigenschaften, der Verhaltensweisen, des Zusammenwirkens und des Aufbaus von (bestehenden oder zu entwickelnden) Modellen, Programmen oder Systemen
- Ziele:

Spezifikation

- Präzise Darstellung und Beschreibung der Eigenschaften, der Verhaltensweisen, des Zusammenwirkens und des Aufbaus von (bestehenden oder zu entwickelnden) Modellen, Programmen oder Systemen
- Ziele:

  - klareres Verständnis des Problems, seiner Eigenschaften und Lösungen

Spezifikation

- Präzise Darstellung und Beschreibung der Eigenschaften, der Verhaltensweisen, des Zusammenwirkens und des Aufbaus von (bestehenden oder zu entwickelnden) Modellen, Programmen oder Systemen
- Ziele:

  - klareres Verständnis des Problems, seiner Eigenschaften und Lösungen
  - (absolutes) Vergleichsmaß, ob ein System den gewünschten Anforderungen genügt

Spezifikation

- Präzise Darstellung und Beschreibung der Eigenschaften, der Verhaltensweisen, des Zusammenwirkens und des Aufbaus von (bestehenden oder zu entwickelnden) Modellen, Programmen oder Systemen
- Ziele:
  - klareres Verständnis des Problems, seiner Eigenschaften und Lösungen
  - (absolutes) Vergleichsmaß, ob ein System den gewünschten Anforderungen genügt
  - beschreibt zusätzliche Aspekte wie Zuverlässigkeit, Robustheit, zeitl. Verhalten

Beweisbare Eigenschaften (1):

Beweisbare Eigenschaften (1):

Partielle Korrektheit:

Ein Programm heißt partiell korrekt, falls es nach einer korrekten Eingabe im Falle der Termination mit einer erwarteten Ausgabe endet.

Beweisbare Eigenschaften (1):

Partielle Korrektheit:
Ein Programm heißt partiell korrekt, falls es nach einer korrekten Eingabe im Falle der Termination mit einer erwarteten Ausgabe endet.

Totale Korrektheit:
Ein Programm heißt total korrekt, falls es nach einer korrekten Eingabe terminiert, und zwar mit einer erwarteten Ausgabe.

Beweisbare Eigenschaften (2):

Beweisbare Eigenschaften (2):

z.B. Parallele Programme:

Beweisbare Eigenschaften (2):

z.B. Parallele Programme:

- Interferenzfreiheit

Beweisbare Eigenschaften (2):

z.B. Parallele Programme:

- Interferenzfreiheit
- Deadlock-Freiheit

Beweisbare Eigenschaften (2):

z.B. Parallele Programme:

- Interferenzfreiheit
- Deadlock-Freiheit
- Korrektheit unter Fairneß-Annahmen

Beweisbare Eigenschaften (2):

z.B. Parallele Programme:

- Interferenzfreiheit
- Deadlock-Freiheit
- Korrektheit unter Fairneß-Annahmen
- ...

Das Hoaresche Beweissystem

Das Hoaresche Beweissystem

- Hoare, 1969

Das Hoaresche Beweissystem

- Hoare, 1969
- Floyd 1967: Flußdiagramme

Das Hoaresche Beweissystem

- Hoare, 1969
- Floyd 1967: Flußdiagramme
- Grundlage: Prädikatenlogik, induktive Zusicherungs-Methode

Das Hoaresche Beweissystem

- Hoare, 1969
- Floyd 1967: Flußdiagramme
- Grundlage: Prädikatenlogik, induktive Zusicherungs-Methode
- WHILE-Programme

Das Hoaresche Beweissystem

- Hoare, 1969
- Floyd 1967: Flußdiagramme
- Grundlage: Prädikatenlogik, induktive Zusicherungs-Methode
- WHILE-Programme

  --> Hoare-Kalkül

Die Sprache:

Die Sprache:

- Leere Anweisung: *skip*

Die Sprache:

- Leere Anweisung: *skip*
- Wertzuweisung: $u := t$

Die Sprache:

- Leere Anweisung: *skip*
- Wertzuweisung: $u := t$
- sequentielle Komposition: $S_1; S_2$

Die Sprache:

- Leere Anweisung: *skip*
- Wertzuweisung: $u := t$
- sequentielle Komposition: $S_1; S_2$
- bedingte Anweisung: ***if*** B ***then*** S_1 ***else*** S_2 ***fi***

Die Sprache:

- Leere Anweisung: *skip*
- Wertzuweisung: $u := t$
- sequentielle Komposition: $S_1 ; S_2$
- bedingte Anweisung: ***if*** B ***then*** S_1 ***else*** S_2 ***fi***
- Schleife: ***while*** B ***do*** S_1 ***od***

Die Semantik:

Die Semantik:

Definition allgemein:

Abbildung von Anfangszuständen in Endzustände:

Die Semantik:

Definition allgemein:

Abbildung von Anfangszuständen in Endzustände:

$$M[\![S]\!]: \Sigma \rightarrow P(\Sigma)$$

Die Semantik:

Definition allgemein:

Abbildung von Anfangszuständen in Endzustände:

$$M[\![S]\!]: \Sigma \rightarrow P(\Sigma)$$

Ansätze:

Die Semantik:

Definition allgemein:

Abbildung von Anfangszuständen in Endzustände:

$$M[\![S]\!]: \Sigma \rightarrow P(\Sigma)$$

Ansätze:

- Übersetzersemantik (Rückführung auf bekannte Programmiersprache)

Die Semantik:

Definition allgemein:

Abbildung von Anfangszuständen in Endzustände:

$$M[\![S]\!]: \Sigma \rightarrow P(\Sigma)$$

Ansätze:

- Übersetzersemantik (Rückführung auf bekannte Programmiersprache)
- denotationelle Semantik (Darstellung der Zustandsänderungen mit Hilfe von Konfigurationen)

Die Semantik:

Definition allgemein:

Abbildung von Anfangszuständen in Endzustände:

$$M[\![S]\!]: \Sigma \rightarrow P(\Sigma)$$

Ansätze:

- Übersetzersemantik (Rückführung auf bekannte Programmiersprache)
- denotationelle Semantik (Darstellung der Zustandsänderungen mit Hilfe von Konfigurationen)
- operationelle Semantik (Darstellung der Zustandsänderungen mit Hilfe von semantischen Funktionen)

Die Semantik:

Definition allgemein:

Abbildung von Anfangszuständen in Endzustände:

$$M[\![S]\!]: \Sigma \rightarrow P(\Sigma)$$

Ansätze:

- Übersetzersemantik (Rückführung auf bekannte Programmiersprache)
- denotationelle Semantik (Darstellung der Zustandsänderungen mit Hilfe von Konfigurationen)
- operationelle Semantik (Darstellung der Zustandsänderungen mit Hilfe von semantischen Funktionen)
- axiomatische Semantik (Darstellung der Auswirkungen von Zustandsänderungen auf die Eigenschaften von Zuständen)

Axiomatischer Ansatz: Hoare-Tripel

Axiomatischer Ansatz: Hoare-Tripel

{p} S {q}

Axiomatischer Ansatz: Hoare-Tripel

{p} S {q}

Bedeutung:

Axiomatischer Ansatz: Hoare-Tripel

{p} S {q}

Bedeutung:

Wenn für das Programm vor der Ausführung der Anweisung S die „Vorbedingung“ p gilt und die Anweisung terminiert, dann gilt danach die „Nachbedingung“ q.

Axiomatischer Ansatz: Hoare-Tripel

{p} S {q}

Bedeutung:

Wenn für das Programm vor der Ausführung der Anweisung S die „Vorbedingung“ p gilt und die Anweisung terminiert, dann gilt danach die „Nachbedingung“ q.

Korrektheit (2)

Korrektheit (2)

{p} S {q} heißt partiell korrekt, wenn jede terminierende Berechnung von S, die in einem p-Zustand startet, in einem q-Zustand terminiert.

Korrektheit (2)

{p} S {q} heißt partiell korrekt, wenn jede terminierende Berechnung von S, die in einem p-Zustand startet, in einem q-Zustand terminiert.

{p} S {q} heißt total korrekt, wenn jede Berechnung von S, die in einem p-Zustand startet, terminiert und ihren Endzustand q erfüllt.

Korrektheit (3)

Korrektheit (3)

Definition:
Eine Korrektheitsformel {p} S {q} gilt im Sinne der partiellen (bzw. totalen) Korrektheit,
$\models\{p\}\ S\ \{q\}$ (bzw. $\models_{tot}\ \{p\}\ S\ \{q\}$),
falls

$$M[\![S]\!]([\![p]\!]) \subseteq [\![q]\!]$$

bzw

$$M_{tot}[\![S]\!]\ ([\![p]\!]) \subseteq [\![q]\!]$$

Das Hoare-Kalkül (1)

Das Hoare-Kalkül (1)

Beweissystem PD:

Das Hoare-Kalkül (1)

Beweissystem PD:

AXIOM 1: Skip-Anweisung

Das Hoare-Kalkül (1)

Beweissystem PD:

AXIOM 1: Skip-Anweisung

$\{p\}\ \text{skip}\ \{q\}$

Das Hoare-Kalkül (1)

Beweissystem PD:

AXIOM 1: Skip-Anweisung

{p} skip {q}

AXIOM 2: Wertzuweisung

Das Hoare-Kalkül (1)

Beweissystem PD:

AXIOM 1: Skip-Anweisung

$$\{p\}\ \text{skip}\ \{q\}$$

AXIOM 2: Wertzuweisung

$$\{p(u:=t)\}\ u:=t\ \{p\}$$

Das Hoare-Kalkül (2)

Das Hoare-Kalkül (2)

REGEL 3: Sequentielle Komposition

Das Hoare-Kalkül (2)

REGEL 3: Sequentielle Komposition

$$\underline{\{p\}\ S_1\ \{r\},\ \{r\}\ S_2\ \{q\}}$$

Das Hoare-Kalkül (2)

REGEL 3: Sequentielle Komposition

$$\frac{\{p\}\ S_1\ \{r\},\ \{r\}\ S_2\ \{q\}}{\{p\}\ S_1;S_2\ \{q\}}$$

Das Hoare-Kalkül (2)

REGEL 3: Sequentielle Komposition

$$\frac{\{p\}\ S_1\ \{r\},\ \{r\}\ S_2\ \{q\}}{\{p\}\ S_1;S_2\ \{q\}}$$

REGEL 4: Bedingte Anweisung:

Das Hoare-Kalkül (2)

REGEL 3: Sequentielle Komposition

$$\frac{\{p\}\ S_1\ \{r\},\ \{r\}\ S_2\ \{q\}}{\{p\}\ S_1;S_2\ \{q\}}$$

REGEL 4: Bedingte Anweisung:

$$\frac{\{p \wedge B\}\ S_1\ \{q\}, \{p \wedge \neg B\}\ S_2\ \{q\}}{}$$

Das Hoare-Kalkül (2)

REGEL 3: Sequentielle Komposition

$$\frac{\{p\}\ S_1\ \{r\},\ \{r\}\ S_2\ \{q\}}{\{p\}\ S_1;S_2\ \{q\}}$$

REGEL 4: Bedingte Anweisung:

$$\frac{\{p \wedge B\}\ S_1\ \{q\}, \{p \wedge \neg B\}\ S_2\ \{q\}}{\{p\}\ \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi } \{q\}}$$

Das Hoare-Kalkül (3)

Das Hoare-Kalkül (3)

REGEL 5: Schleife:

Das Hoare-Kalkül (3)

REGEL 5: Schleife:

$$\frac{\{p \wedge B\}\ S\ \{p\}}{}$$

Das Hoare-Kalkül (3)

REGEL 5: Schleife:

$$\frac{\{p \wedge B\}\ S\ \{p\}}{\{p\}\ \text{while } B \text{ do } S \text{ od } \{p \wedge \neg B\}}$$

Das Hoare-Kalkül (3)

REGEL 5: Schleife:

$$\frac{\{p \wedge B\}\ S\ \{p\}}{\{p\}\ \text{while } B \text{ do } S \text{ od } \{p \wedge \neg B\}}$$

REGEL 6: Konsequenzregel:

Das Hoare-Kalkül (3)

REGEL 5: Schleife:

$$\frac{\{p \wedge B\}\ S\ \{p\}}{\{p\}\ \text{while } B \text{ do } S \text{ od } \{p \wedge \neg B\}}$$

REGEL 6: Konsequenzregel:

$$\frac{p \rightarrow p_1,\ \{p_1\}\ S\ \{q_1\},\ q_1 \rightarrow q}{}$$

Das Hoare-Kalkül (3)

REGEL 5: Schleife:

$$\frac{\{p \wedge B\}\ S\ \{p\}}{\{p\}\ \text{while } B \text{ do } S \text{ od } \{p \wedge \neg B\}}$$

REGEL 6: Konsequenzregel:

$$\frac{p \rightarrow p_1,\ \{p_1\}\ S\ \{q_1\},\ q_1 \rightarrow q}{\{p\}\ S\ \{q\}}$$

Beispiele zur Anwendung des Beweissystems PD (1):

Beispiele zur Anwendung des Beweissystems PD (1):

$S \equiv x:=x+1;\ y:=y+1$

Beispiele zur Anwendung des Beweissystems PD (2):

Beispiele zur Anwendung des Beweissystems PD (2):

$DIV \equiv quo:=0;\ rem:=x;\ S_0,$

Beispiele zur Anwendung des Beweissystems PD (2):

DIV ≡ quo:=0; rem:= x; S_0,
wobei S_0 ≡ while rem ⩾ y do rem:=rem-y;
quo:=quo+1 od,

Beispiele zur Anwendung des Beweissystems PD (2):

DIV ≡ quo:=0; rem:= x; S_0,
wobei S_0 ≡ while rem ⩾ y do rem:=rem-y;
quo:=quo+1 od,

Das Hoare-Kalkül (4)

Das Hoare-Kalkül (4)

Totale Korrektheit:

Das Hoare-Kalkül (4)

Totale Korrektheit:

REGEL 7: Schleife II:

Das Hoare-Kalkül (4)

Totale Korrektheit:

REGEL 7: Schleife II:

$\{p \wedge B\}\ S\ \{p\}$,

Das Hoare-Kalkül (4)

Totale Korrektheit:

REGEL 7: Schleife II:

$$\{p \wedge B\}\ S\ \{p\},$$

$$\{p \wedge B \wedge t = z\}\ S\ \{t<z\},$$

Das Hoare-Kalkül (4)

Totale Korrektheit:

REGEL 7: Schleife II:

$$\frac{\{p \wedge B\}\ S\ \{p\},\quad \{p \wedge B \wedge t = z\}\ S\ \{t<z\},\quad p \rightarrow t \geq 0,}{}$$

Das Hoare-Kalkül (4)

Totale Korrektheit:

REGEL 7: Schleife II:

$$\frac{\{p \wedge B\}\ S\ \{p\},\quad \{p \wedge B \wedge t = z\}\ S\ \{t<z\},\quad p \rightarrow t \geq 0,}{\{p\}\ \text{while } B \text{ do } S \text{ od } \{p \wedge \neg B\}}$$

Das Hoare-Kalkül (4)

Totale Korrektheit:

REGEL 7: Schleife II:

$$\frac{\{p \wedge B\}\ S\ \{p\},\quad \{p \wedge B \wedge t = z\}\ S\ \{t<z\},\quad p \rightarrow t \geq 0,}{\{p\}\ \text{while } B \text{ do } S \text{ od } \{p \wedge \neg B\}}$$

(wobei t int-Ausdruck; z int-Variable, die nicht in p, B, t, S vorkommt)

Das Hoare-Kalkül (5)

Das Hoare-Kalkül (5)

Totale Korrektheit:

Das Hoare-Kalkül (5)

Totale Korrektheit:

Beweissystem TD:

Das Hoare-Kalkül (5)

Totale Korrektheit:

Beweissystem TD:

AXIOME 1,2 und REGELN 3, 4, 6, 7

Beispiele zur Anwendung des Beweissystems PD (3):

DIV ≡ quo:=0; rem:= x; S_0,
wobei S_0 ≡ while rem ⩾ y do rem:=rem-y;
quo:=quo+1 od,

Eigenschaften von PD und TD

Eigenschaften von PD und TD

- Korrektheit

Eigenschaften von PD und TD

- Korrektheit
- Vollständigkeit

Systematische Entwicklung korrekter Programme (1)

Systematische Entwicklung korrekter Programme (1)

- Dijkstra (19??)

Systematische Entwicklung korrekter Programme (1)

- Dijkstra (19??)
- Regeln aus den Regeln der Programmverifikation

abgeleitet

z.B. Spezifikation von Problemen in der Form

{p} S {q}

Systematische Entwicklung korrekter Programme (1)

- Dijkstra (19??)
- Regeln aus den Regeln der Programmverifikation

abgeleitet

z.B. Spezifikation von Problemen in der Form

{p} S {q}

- syst. Entwicklung von Schleifen

S≡T; **while** B **do** R **od**

- Schleifeninvariante -

Systematische Entwicklung korrekter Programme (2)

Systematische Entwicklung korrekter Programme (2)

Allgemeines Vorgehen:

Systematische Entwicklung korrekter Programme (2)

Allgemeines Vorgehen:

1. Beschreiben des funktionalen Verhaltens eines Programmes: formale Spezifikation

Systematische Entwicklung korrekter Programme (2)

Allgemeines Vorgehen:

1. Beschreiben des funktionalen Verhaltens eines Programmes: formale Spezifikation
2. Beschreibung der Sicherheitsanforderungen

Systematische Entwicklung korrekter Programme (2)

Allgemeines Vorgehen:

1. Beschreiben des funktionalen Verhaltens eines Programmes: formale Spezifikation
2. Beschreibung der Sicherheitsanforderungen
3. Nachweis der Sicherheit und Validierung von Spezifikationen

Systematische Entwicklung korrekter Programme (2)

Allgemeines Vorgehen:

1. Beschreiben des funktionalen Verhaltens eines Programmes: formale Spezifikation
2. Beschreibung der Sicherheitsanforderungen
3. Nachweis der Sicherheit und Validierung von Spezifikationen
4. Implementierung und deren Verifikation

Formale Programmverifikation

- Einführung/Motivation
- Softwareverifikation am Beispiel des Hoare-Kalküls
- Überblick über verschiedene Verifikationsmethoden
- Rechnerunterstützte Verifikationsmethoden
- Möglichkeiten und Grenzen der SV

Weiterentwicklung des Hoare-Kalküls (1)

Weiterentwicklung des Hoare-Kalküls (1)

AXIOM A1: Invarianz:

Weiterentwicklung des Hoare-Kalküls (1)

AXIOM A1: Invarianz:

$$\{p\}\ S\ \{p\}$$

Weiterentwicklung des Hoare-Kalküls (1)

AXIOM A1: Invarianz:

$$\{p\}\ S\ \{p\}$$

REGEL A2: Disjunktion:

Weiterentwicklung des Hoare-Kalküls (1)

AXIOM A1: Invarianz:

$$\{p\}\ S\ \{p\}$$

REGEL A2: Disjunktion:

$$\frac{\{p\}\ S\ \{q\},\ \{r\}\ S\ \{q\}}{}$$

Weiterentwicklung des Hoare-Kalküls (1)

AXIOM A1: Invarianz:

$$\{p\}\ S\ \{p\}$$

REGEL A2: Disjunktion:

$$\frac{\{p\}\ S\ \{q\},\ \{r\}\ S\ \{q\}}{\{p \vee r\}\ S\ \{q\}}$$

Weiterentwicklung des Hoare-Kalküls (1)

AXIOM A1: Invarianz:

$$\{p\}\ S\ \{p\}$$

REGEL A2: Disjunktion:

$$\frac{\{p\}\ S\ \{q\},\ \{r\}\ S\ \{q\}}{\{p \vee r\}\ S\ \{q\}}$$

REGEL A3: $\exists$-Einführung:

Weiterentwicklung des Hoare-Kalküls (1)

AXIOM A1: Invarianz:

$$\{p\}\ S\ \{p\}$$

REGEL A2: Disjunktion:

$$\frac{\{p\}\ S\ \{q\},\ \{r\}\ S\ \{q\}}{\{p \vee r\}\ S\ \{q\}}$$

REGEL A3: ∃-Einführung:

$$\frac{\{p\}\ S\ \{q\}}{}$$

Weiterentwicklung des Hoare-Kalküls (1)

AXIOM A1: Invarianz:

$$\{p\}\ S\ \{p\}$$

REGEL A2: Disjunktion:

$$\frac{\{p\}\ S\ \{q\},\ \{r\}\ S\ \{q\}}{\{p \vee r\}\ S\ \{q\}}$$

REGEL A3: ∃-Einführung:

$$\frac{\{p\}\ S\ \{q\}}{\{\exists x: p\}\ S\ \{q\}}$$

Weiterentwicklung des Hoare-Kalküls (1)

AXIOM A1: Invarianz:

$$\{p\}\ S\ \{p\}$$

REGEL A2: Disjunktion:

$$\frac{\{p\}\ S\ \{q\},\ \{r\}\ S\ \{q\}}{\{p \vee r\}\ S\ \{q\}}$$

REGEL A3: ∃-Einführung:

$$\frac{\{p\}\ S\ \{q\}}{\{\exists x: p\}\ S\ \{q\}}$$

(...)

Weiterentwicklung des Hoare-Kalküls (2)

Weiterentwicklung des Hoare-Kalküls (2)

- andere deterministische Programme

Weiterentwicklung des Hoare-Kalküls (2)

- andere deterministische Programme
- parallele Programme

Weiterentwicklung des Hoare-Kalküls (2)

- andere deterministische Programme
- parallele Programme
- nichtdeterministische Programme

Weiterentwicklung des Hoare-Kalküls (2)

- andere deterministische Programme
- parallele Programme
- nichtdeterministische Programme
- verteilte Programme

Beweissysteme - Ansätze:

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational
  - algebraisch

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational
  - algebraisch
  - ...

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational
  - algebraisch
  - ...
- Unterschiedliche Logiken

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational
  - algebraisch
  - ...
- Unterschiedliche Logiken
  - Prädikaten-Logik

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational
  - algebraisch
  - ...
- Unterschiedliche Logiken
  - Prädikaten-Logik
  - algorithmische Logik

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational
  - algebraisch
  - ...
- Unterschiedliche Logiken
  - Prädikaten-Logik
  - algorithmische Logik
  - dynamische Logik

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational
  - algebraisch
  - ...
- Unterschiedliche Logiken
  - Prädikaten-Logik
  - algorithmische Logik
  - dynamische Logik
  - LCF-Logik

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational
  - algebraisch
  - ...
- Unterschiedliche Logiken
  - Prädikaten-Logik
  - algorithmische Logik
  - dynamische Logik
  - LCF-Logik
  - ...

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational
  - algebraisch
  - ...
- Unterschiedliche Logiken
  - Prädikaten-Logik
  - algorithmische Logik
  - dynamische Logik
  - LCF-Logik
  - ...
- Rechnerunterstützte Verifikationsmethoden

Beweissysteme - Ansätze:

- Unterschiedliche Semantiken:
  - operational
  - denotational
  - algebraisch
  - ...
- Unterschiedliche Logiken
  - Prädikaten-Logik
  - algorithmische Logik
  - dynamische Logik
  - LCF-Logik
  - ...
- Rechnerunterstützte Verifikationsmethoden
- ...

Formale Programmverifikation

- Einführung/Motivation
- Softwareverifikation am Beispiel des Hoare-Kalküls
- Überblick über verschiedene Verifikationsmethoden
- Rechnerunterstützte Verifikationsmethoden
- Möglichkeiten und Grenzen der SV

Rechnerunterstützte Verifikationsmethoden

Rechnerunterstützte Verifikationsmethoden

- KIV

Rechnerunterstützte Verifikationsmethoden

- KIV
- LCF-System

Rechnerunterstützte Verifikationsmethoden

- KIV
- LCF-System
- Stanford-Verifier

Rechnerunterstützte Verifikationsmethoden

- KIV
- LCF-System
- Stanford-Verifier
- PL/CV2 Proof Checker

Rechnerunterstützte Verifikationsmethoden

- KIV
- LCF-System
- Stanford-Verifier
- PL/CV2 Proof Checker
- Boyer-Moore-System

Rechnerunterstützte Verifikationsmethoden

- KIV
- LCF-System
- Stanford-Verifier
- PL/CV2 Proof Checker
- Boyer-Moore-System
- ...

Rechnerunterstützte Verifikationsmethoden

- KIV
- LCF-System
- Stanford-Verifier
- PL/CV2 Proof Checker
- Boyer-Moore-System
- ...

$\longrightarrow$ Dialogsysteme

KIV

KIV

- Unterstützungswerkzeug zur Entwicklung korrekter Software

KIV

- Unterstützungswerkzeug zur Entwicklung korrekter Software
- graphische Oberfläche

KIV

- Unterstützungswerkzeug zur Entwicklung korrekter Software
- graphische Oberfläche
- unterstützt:

KIV

- Unterstützungswerkzeug zur Entwicklung korrekter Software
- graphische Oberfläche
- unterstützt:
    - Spezifikationsentwicklung

KIV

- Unterstützungswerkzeug zur Entwicklung korrekter Software
- graphische Oberfläche
- unterstützt:
    - Spezifikationsentwicklung
    - Nachweis von Sicherheitseigenschaften

KIV

- Unterstützungswerkzeug zur Entwicklung korrekter Software
- graphische Oberfläche
- unterstützt:
    - Spezifikationsentwicklung
    - Nachweis von Sicherheitseigenschaften
    - Implementierung

KIV

- Unterstützungswerkzeug zur Entwicklung korrekter Software
- graphische Oberfläche
- unterstützt:
    - Spezifikationsentwicklung
    - Nachweis von Sicherheitseigenschaften
    - Implementierung
    - Verifikation

KIV - Spezifikationsentwicklung

KIV - Spezifikationsentwicklung

- Aufbauend auf element. algebr. Spezifikationen können große modulare Spezifikationen erstellt werden

KIV - Spezifikationsentwicklung

- Aufbauend auf element. algebr. Spezifikationen können große modulare Spezifikationen erstellt werden
- Automat. Überprüfung der syntakt. Korrektheit

KIV - Spezifikationsentwicklung

- Aufbauend auf element. algebr. Spezifikationen können große modulare Spezifikationen erstellt werden
- Automat. Überprüfung der syntakt. Korrektheit
- Visualisierung als Entwicklungsgraphen, darüber Bearbeitung, Verwaltung möglich

KIV - Spezifikationsentwicklung

- Aufbauend auf element. algebr. Spezifikationen können große modulare Spezifikationen erstellt werden
- Automat. Überprüfung der syntakt. Korrektheit
- Visualisierung als Entwicklungsgraphen, darüber Bearbeitung, Verwaltung möglich
- große Bibliothek wiederverwendbarer Standardspezifikationen

KIV- Beweisen in Spezifikationen

KIV- Beweisen in Spezifikationen

- Weitgehend automatisiert, jedoch an vielen Stellen Interaktion des Benutzers erforderlich

KIV- Beweisen in Spezifikationen

- Weitgehend automatisiert, jedoch an vielen Stellen Interaktion des Benutzers erforderlich
- Beweisgang graphisch visualisiert; darüber Verwaltung, Modifikation, Dokumentation möglich

KIV- Beweisen in Spezifikationen

- Weitgehend automatisiert, jedoch an vielen Stellen Interaktion des Benutzers erforderlich
- Beweisgang graphisch visualisiert; darüber Verwaltung, Modifikation, Dokumentation möglich
- Korrektheitsmanagement (verwaltet Beweispflichten, verhindert z.B. zyklisches verwenden von Lemmata)

KIV- Implementierung & Verifikation

KIV- Implementierung & Verifikation

- Korrektheitsmanagement

KIV- Implementierung & Verifikation

- Korrektheitsmanagement
- Beweiskomponente (zu 80-90% automatisch)

KIV- Implementierung & Verifikation

- Korrektheitsmanagement
- Beweiskomponente (zu 80-90% automatisch)
- Beweisstrategie ergänzt um spezielle Beweisregeln für Programme

KIV- Implementierung & Verifikation

- Korrektheitsmanagement
- Beweiskomponente (zu 80-90% automatisch)
- Beweisstrategie ergänzt um spezielle Beweisregeln für Programme
- Automatische Löschung der von Korrekturen betroffenen Beweise

KIV- Implementierung & Verifikation

- Korrektheitsmanagement
- Beweiskomponente (zu 80-90% automatisch)
- Beweisstrategie ergänzt um spezielle Beweisregeln für Programme
- Automatische Löschung der von Korrekturen betroffenen Beweise
- Strategie zur Wiederverwendung früherer Beweise (Ersparnis von ~90% des Aufwands)

KIV- Anwendungen

KIV- Anwendungen

- Zusammenarbeit mit über 10 Behörden, Institutionen, Firmen

KIV- Anwendungen

- Zusammenarbeit mit über 10 Behörden, Institutionen, Firmen
- Pilotanwendungen in:

KIV- Anwendungen

- Zusammenarbeit mit über 10 Behörden, Institutionen, Firmen
- Pilotanwendungen in: • Raumfahrt

KIV- Anwendungen

- Zusammenarbeit mit über 10 Behörden, Institutionen, Firmen
- Pilotanwendungen in:
  - Raumfahrt
  - Medizintechnik

KIV- Anwendungen

- Zusammenarbeit mit über 10 Behörden, Institutionen, Firmen
- Pilotanwendungen in:
  - Raumfahrt
  - Medizintechnik
  - Automobiltechnik

KIV- Anwendungen

- Zusammenarbeit mit über 10 Behörden, Institutionen, Firmen
- Pilotanwendungen in:
  - Raumfahrt
  - Medizintechnik
  - Automobiltechnik
  - Bahntechnik

KIV- Anwendungen

- Zusammenarbeit mit über 10 Behörden, Institutionen, Firmen
- Pilotanwendungen in:
  - Raumfahrt
  - Medizintechnik
  - Automobiltechnik
  - Bahntechnik
- Crash-Control-Software für Airbag

KIV- Anwendungen

- Zusammenarbeit mit über 10 Behörden, Institutionen, Firmen
- Pilotanwendungen in:
  - Raumfahrt
  - Medizintechnik
  - Automobiltechnik
  - Bahntechnik
- Crash-Control-Software für Airbag
- Compilerverifikation

KIV- Anwendungen

- Zusammenarbeit mit über 10 Behörden, Institutionen, Firmen
- Pilotanwendungen in:
  - Raumfahrt
  - Medizintechnik
  - Automobiltechnik
  - Bahntechnik
- Crash-Control-Software für Airbag
- Compilerverifikation
- ...

Formale Programmverifikation

- Einführung/Motivation
- Softwareverifikation am Beispiel des Hoare-Kalküls
- Überblick über verschiedene Verifikationsmethoden
- Rechnerunterstützte Verifikationsmethoden
- Möglichkeiten und Grenzen der SV

Programmverifikation zur Qualitätssicherung?

- Einsetzbarkeit in der Praxis?
- Notwendigkeit?
- Aufwand?
- Kosten?
- Handhabbarkeit?
- Vorteile gegenüber anderen Methoden?

Probleme

Probleme

- Es kann nur bewiesen werden, was auch spezifiziert wurde!

Probleme

- Es kann nur bewiesen werden, was auch spezifiziert wurde!
- v.a. bei größeren Programmen sehr aufwendig (-> teuer?)

Probleme

- Es kann nur bewiesen werden, was auch spezifiziert wurde!
- v.a. bei größeren Programmen sehr aufwendig (-> teuer?)
- spezielle logische u. mathematische Kenntnisse erforderlich (-> Spezialisten?)

Probleme

- Es kann nur bewiesen werden, was auch spezifiziert wurde!
- v.a. bei größeren Programmen sehr aufwendig (-> teuer?)
- spezielle logische u. mathematische Kenntnisse erforderlich (-> Spezialisten?)
- Robustheit, Zuverlässigkeit, zeitl. Verhalten schwer formalisierbar

ABER:

ABER:

- Es wird stets die Korrektheit für ALLE Eingaben bewiesen (-> Gewißheit)

ABER:

- Es wird stets die Korrektheit für ALLE Eingaben bewiesen (-> Gewißheit)
- frühzeitige Aufdeckung von Fehlern möglich (-> erspart Zeit und Geld)

ABER:

- Es wird stets die Korrektheit für ALLE Eingaben bewiesen (-> Gewißheit)
- frühzeitige Aufdeckung von Fehlern möglich (-> erspart Zeit und Geld)
- Mittlerweile gute rechnerunterstützte Verfahren

ABER:

- Es wird stets die Korrektheit für ALLE Eingaben bewiesen (-> Gewißheit)
- frühzeitige Aufdeckung von Fehlern möglich (-> erspart Zeit und Geld)
- Mittlerweile gute rechnerunterstützte Verfahren
- Kooperation mit Universitäten möglich

ABER:

- Es wird stets die Korrektheit für ALLE Eingaben bewiesen (-> Gewißheit)
- frühzeitige Aufdeckung von Fehlern möglich (-> erspart Zeit und Geld)
- Mittlerweile gute rechnerunterstützte Verfahren
- Kooperation mit Universitäten möglich
- gerade bei kritischen Anwendungen Softwareverifikation unabdingbar

These:

Softwareverifikation ist eine

Frage der Qualität!

Programmverifikation!