Programmverifikation

http://www.informatik.uni-bremen.de/theorie/teach/veri

Renate Klempien-Hinrichs

- Wer sind wir?
- Wie ist der Kurs organisiert?
- Worum geht es?

Wer sind wir?

Renate Klempien-Hinrichs

OAS 30??

Tel. 218-8791

kh@biba.uni-bremen.de

Sprechstunde: Dienstag 11–13 Uhr

Caro von Totth

OAS 3006

Tel. 218-8792

caro@informatik.uni-bremen.de

Materialien

- **Webseite:**

  http://www.informatik.uni-bremen.de/theorie/teach/veri

  - **Folien**
  - **Übungsblätter**

- **Lehrbuch:**

  Krzysztof R. Apt, Ernst-Rüdiger Olderog:
  Programmverifikation. Sequentielle, parallele und verteilte Programme.
  Springer, 1994.

Termine

- Montag 8–12 MZH 7250 bzw. 7210

  Beginn: pünktlich 8:15 Uhr;
  zwischendrin dreimal ca. 10 Minuten Pause

- 2–3 Nachholtermine in der Woche nach Lehrveranstaltungsende (23.–27.7.2007)

- Fachgespräche / mündliche Modulprüfungen in den zwei Wochen nach den Nachholterminen

Prüfungsleistungen

- "**Schein**", beschlossene Bedingungen:
  - Gruppen à 3–4 Personen
  - wöchentlich 1 Übungsblatt (Freitag 15 Uhr auf der Webseite)
    - Bearbeitungszeit 1 Woche
    - Abgabe bis vor der LV (Montag 8:15 Uhr)
    - $n - 1$ Übungsblätter sind abzugeben
  - Fachgespräch in der Gruppe
  - je Übungsblatt 2 Übungsaufgaben
    - Aufgabe 1 (Grundkenntnisse) muss richtig ($\geq 90\%$) gelöst werden
      - Rücksprache bei Problemen
      - Aufgaben 1 + Fachgespräch = 4,0
    - Aufgabe 2 ist Grundlage für Noten besser als 4,0
  - Gruppenarbeit wird mitbewertet
- **mündliche Modulprüfung** (DPO'03): ja

Programmverifikation

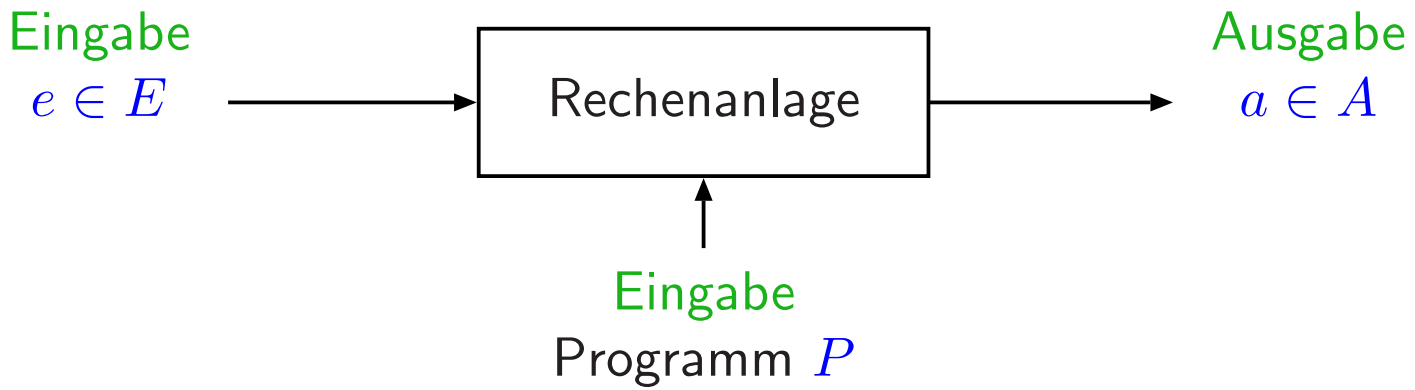
Nachweis, dass ein Programm das tut, was es soll

"Korrektheit"

Wie geht das?

Programm und Verifikation

Programm: Formulierung eines Algorithmus und der zugehörigen Datenbereiche in einer Programmiersprache



Verifikation: formaler Nachweis von Eigenschaften von Programmen oder Programmteilen

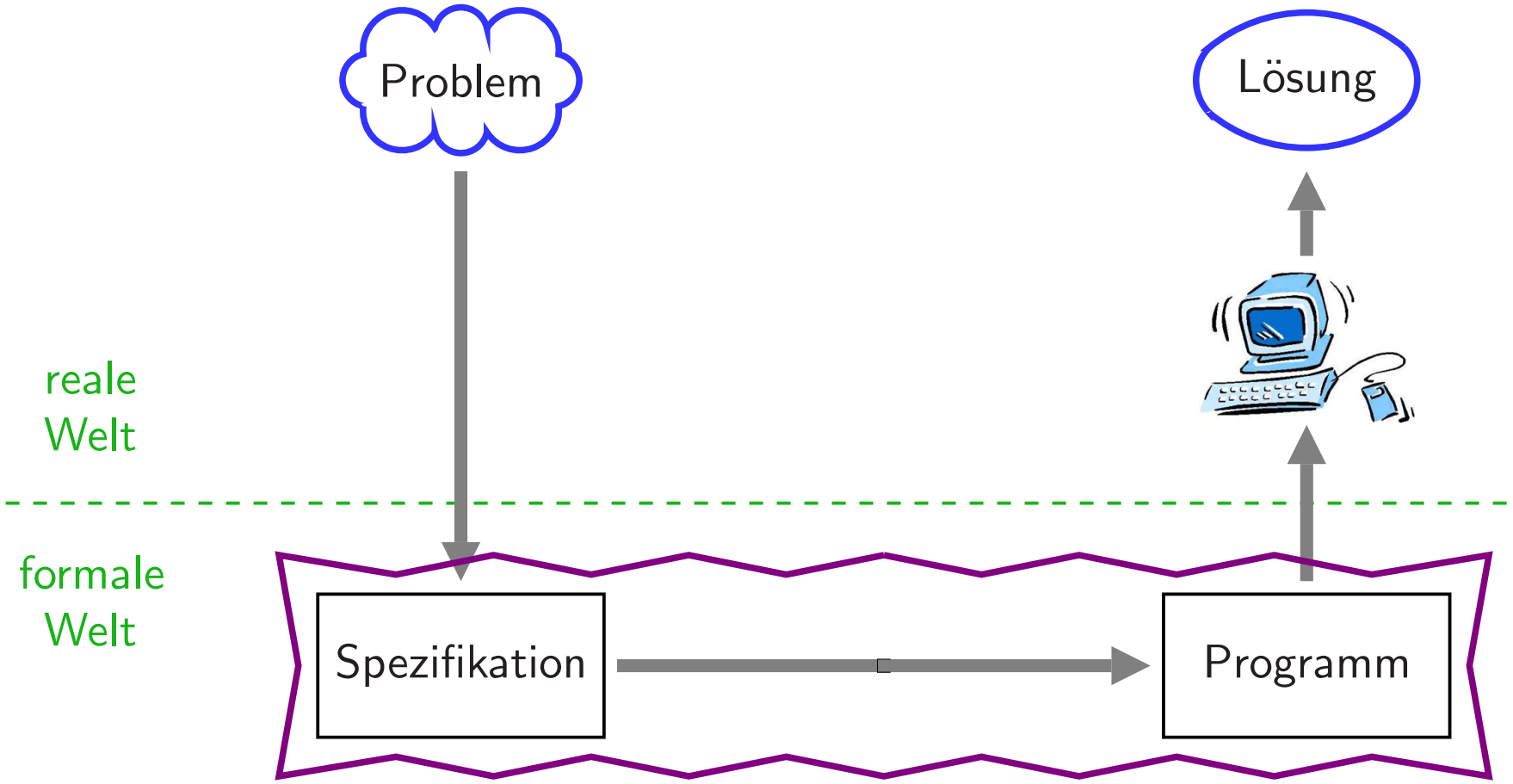
Zentrales Problem

Löst das Programm die gestellte Aufgabe?

"Every program performs *some* task correctly. What is of interest to computer scientists is whether a program performs its *intended* task."

B. Liskov, V. Berzins

Situation



Korrektheitsproblem:
Erfüllt das Programm seine Spezifikation?

Korrektheitseigenschaften

Sei S eine Spezifikation und P ein Programm.

- Partielle Korrektheit:

  Wenn P für eine Eingabe hält, so ist das Ergebnis im Sinne von S richtig.

- Termination:

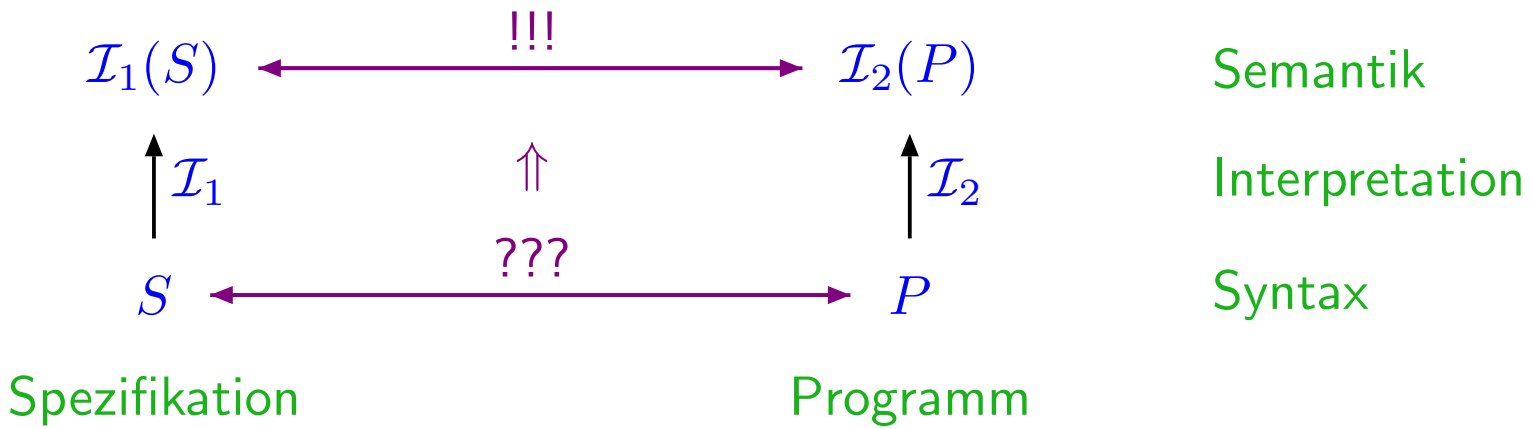
  P hält für jede vorgesehene Eingabe.

- Totale Korrektheit:

  Partielle Korrektheit + Termination

Testen und Verifikation

- **Testen** kann Fehler aufdecken, aber i.a. nicht Korrektheit garantieren.
- **Programmverifikation** ist der systematische Nachweis von Korrektheit mit Hilfe formaler (d.h. mathematischer) Methoden.
- Verifikation setzt voraus, dass
  - Spezifikationen und Programme in Sprachen mit präzisen **Semantiken** vorliegen und
  - eine **mathematische Theorie** über Korrektheitsbeziehungen und Beweismethoden zur Verfügung steht.



Beispiel: Entwicklung eines parallelen Programms

Problem: Nullstellensuche

Gegeben: Funktion $f : \mathbb{Z} \to \mathbb{Z}$, die eine Nullstelle besitzt (d.h. es gibt $k \in \mathbb{Z}$ mit $f(k) = 0$).

Gesucht: Programm $ZERO$, das eine Nullstelle findet.

Lösungsidee: Gleichzeitiges Suchen nach einer positiven $(k > 0)$ und einer nicht-positiven $(k \leq 0)$ Nullstelle.

Verwende dabei die *parallele Komposition* $[S_1||S_2]$ zweier Programme S_1 und S_2. Das Programm $[S_1||S_2]$ wird ausgeführt, indem die Anweisungen in S_1 und S_2 nebeneinander ausgeführt werden. Es terminiert, wenn sowohl S_1 als auch S_2 terminiert.

Lösung 1

$$
\begin{array}{l}
S_1 \equiv found := \textbf{false}; \\
\quad x := 0; \\
\quad \textbf{while } \neg found \textbf{ do} \\
\quad\quad x := x + 1; \\
\quad\quad found := f(x) = 0 \\
\quad \textbf{od}
\end{array}
$$

$$
\begin{array}{l}
S_2 \equiv found := \textbf{false}; \\
\quad y := 1; \\
\quad \textbf{while } \neg found \textbf{ do} \\
\quad\quad y := y - 1; \\
\quad\quad found := f(y) = 0 \\
\quad \textbf{od}
\end{array}
$$

$$ZERO\text{-}1 \equiv [S_1 || S_2]$$

Lösung 2

$$S_1 \equiv x := 0;$$
$$\quad \textbf{while } \neg found \textbf{ do}$$
$$\quad\quad x := x + 1;$$
$$\quad\quad found := f(x) = 0$$
$$\quad \textbf{od}$$

$$S_2 \equiv y := 1;$$
$$\quad \textbf{while } \neg found \textbf{ do}$$
$$\quad\quad y := y - 1;$$
$$\quad\quad found := f(y) = 0$$
$$\quad \textbf{od}$$

$$ZERO\text{-}2 \equiv found := \textbf{false};$$
$$[S_1 || S_2]$$

Lösung 3

$S_1 \equiv x := 0;$
while $\neg found$ **do**
$x := x + 1;$
if $f(x) = 0$ **then**
$found :=$ **true**
fi
od

$S_2 \equiv y := 1;$
while $\neg found$ **do**
$y := y - 1;$
if $f(y) = 0$ **then**
$found :=$ **true**
fi
od

$ZERO\text{-}3 \equiv found :=$ **false**;
$[S_1||S_2]$

Lösung 4

$S_1 \equiv x := 0;$
while $\neg found$ **do**
await $turn = 1$ **then**
$turn := 2$
end;
$x := x + 1;$
if $f(x) = 0$ **then**
$found :=$ **true**
fi
od

$S_2 \equiv y := 1;$
while $\neg found$ **do**
await $turn = 2$ **then**
$turn := 1$
end;
$y := y - 1;$
if $f(y) = 0$ **then**
$found :=$ **true**
fi
od

$ZERO\text{-}4 \equiv turn := 1;$
$found :=$ **false**;
$[S_1 || S_2]$

Lösung 5

$S_1 \equiv x := 0;$
while $\neg found$ **do**
await $turn = 1$ **then**
$turn := 2$
end;
$x := x + 1;$
if $f(x) = 0$ **then**
$found :=$ **true**
fi
od;
$turn := 2$

$S_2 \equiv y := 1;$
while $\neg found$ **do**
await $turn = 2$ **then**
$turn := 1$
end;
$y := y - 1;$
if $f(y) = 0$ **then**
$found :=$ **true**
fi
od;
$turn := 1$

$ZERO\text{-}5 \equiv turn := 1;$
$found :=$ **false**;
$[S_1 || S_2]$

Übersicht

- Organisatorisches, Einführung
- Syntax von deterministischen Programmen
- Semantik von deterministischen Programmen
- Eigenschaften von deterministischen Programmen
- Spezifikation
- Beweisen partieller Korrektheit
- Beweisen totaler Korrektheit
- Vollständigkeit der Beweissysteme
- Syntax und Semantik von disjunkten parallelen Programmen
- Beweisen der Korrektheit von disjunkten parallelen Programmen
- Überblick und Ausblick