Kapitel L: V

V. Erweiterungen und Anwendungen zur Logik

- Produktionsregelsysteme
- Inferenz für Produktionsregelsysteme
- Produktionsregelsysteme mit Negation
- Nicht-monotones Schließen
- Logik und abstrakte Algebren
- Verifikation
- Verifikation mit dem Hoare-Kalkül
- Hoare-Regeln und partielle Korrektheit
- Terminierung

Bemerkungen:

- Literatur zu diesem Kapitel findet sich online:
  „Semantics with Applications“ von Hanne Riis Nielson und Flemming Nielson.
- Eine neuere Version dieses Buches findet man bei Springer:
  „Semantics with Applications: An Appetizer“ von Hanne Riis Nielson und Flemming Nielson.

Verifikation

Software-Qualität

There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.

[C.A.R. Hoare, 1980]

Verifikation

Software-Qualität

There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.

[C.A.R. Hoare, 1980]

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 (?)]

Verifikation

Software-Qualität

There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.

[C.A.R. Hoare, 1980]

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 deMarco, 1991]

Verifikation

Anwendungsbedarf

- Informationssicherheit (Security)
  - Vertraulichkeit
  - Integrität
  - Authentizität
  - Nicht-Rückweisbarkeit (Signaturgesetz)

  Zertifizierung von IT-Systemen durch das Bundesamt für Sicherheit in der Informationstechnik:

  Höhere Stufen der Vertrauenswürdigkeit erfordern formale Spezifikation und formale Verifikation.

  Beispiele
  - Home Banking
  - Geld- und Chipkarten

Verifikation

Anwendungsbedarf (Fortsetzung)

- Systemsicherheit (Safety)

  Software für sicherheitskritische Systeme ist formal zu spezifizieren und zu verifizieren.

  Beispiele:
  Eingebettete Systeme (Embedded Systems) als Regelungssysteme / reaktive Systeme unter Berücksichtigung von Realzeitaspekten in
  - Autos,
  - Flugzeugen,
  - Raumfahrzeugen,
  - Anlagensteuerungen.

Verifikation

Testen als Alternative

Tests können die *Anwesenheit* von Fehlern beweisen, aber nie die *Abwesenheit* von Fehlern (bei unendlich vielen möglichen Eingaben).

Klassifikation von Testverfahren:

- Schnittstellentest (Blackbox-Test)

  Die Ein- / Ausgaberelation wird auf Übereinstimmung mit der Spezifikation geprüft.

- Programmabhängiger Test (Whitebox-Test)

  Möglichst große Teile aller Pfade durch das Programm werden getestet. Eine möglichst große Überdeckung (des Programmcodes) ist erwünscht.

Verifikation

Testen als Alternative (Fortsetzung)

Systematische Auswahl von Testfällen:

- Schnittstellentest

  Pro spezifizierter Bedingung mindestens einen Testfall prüfen, Randbereiche (ggf. von beiden Seiten) prüfen, Maximal-, Minmalwerte nicht vergessen, eine genügend große Anzahl von Normalfällen prüfen.

- Überdeckungstest

  Erwünscht, aber kaum machbar ist eine Wegüberdeckung d.h. jeder Weg wird mindestens einmal durchlaufen.
  Auf jeden Fall nötig ist eine Anweisungsüberdeckung, d.h. jede Anweisung wird mindestens einmal durchlaufen.

Hauptproblem des Testens:

Kombinatorische Explosion der Möglichkeiten für Testfälle

Verifikation

Aufgaben bei der Softwareentwicklung

... unter anderem:

- Spezifikation

  Was soll die Software eigentlich leisten?

- Implementierung

  Wie soll etwas gemacht werden?

- Korrektheitsprüfung

  Tut das Programm auch das, was es soll?

- Komplexitätsuntersuchung

  Wie gut ist das Programm eigentlich (Zeit, Platz, Struktur)?

Verifikation

Korrektheitsprüfung: Softwaretest

Fehler

- syntaktische Fehler
- semantische Fehler
- Terminierungsfehler

Methoden

- Test
  Überprüfung der Korrektheit einer Implementierung für endlich viele Eingaben
  ➜ Dynamischer Test

- Verifikation
  Nachweis, dass eine Implementierung fehlerfrei das macht, was eine Spezifikation vorschreibt.
  ➜ Statischer Test

Algorithmus und Programm

Algorithmus:

Unter einem Algorithmus versteht man eine Verarbeitungsvorschrift, die so präzise formuliert ist, dass sie von einem mechanisch oder elektronisch arbeitenden Gerät durchgeführt werden kann. Aus der Präzision der sprachlichen Darstellung des Algorithmus muss die Abfolge der einzelnen Verarbeitungsschritte eindeutig hervorgehen.

. . .

[Duden - Informatik]

Programm

Formulierung eines Algorithmus und der zugehörigen Datenbereiche in einer Programmiersprache.

Während Algorithmen relativ allgemein beschrieben werden können und an keine formellen Vorschriften gebunden sind, müssen Programme wesentlich konkreter sein:

- *Sie sind im exakt definierten und eindeutigen Formalismus einer Programmiersprache verfasst.*
- *Sie nehmen Bezug auf eine bestimmte Darstellung der verwendeten Daten.*
- *Sie sind auf einer Rechenanlage ausführbar.*

Ein und derselbe Algorithmus kann in verschiedenen Programmiersprachen formuliert werden; er bildet eine Abstraktion aller Programme, die ihn beschreiben.

[Duden - Informatik]

Verifikation

Arten von Programmierparadigmen

- imperativ (prozedural) z.B. Pascal, C

```
int fibonacci(int x) {
  if ( x == 0) return 1;
  if ( x == 1) return 1;
  return fibonacci(x-1) + fibonacci(x-2);
}
```

- funktional z.B. Lisp

```
(defun fibonacci (schrittzahl)
  (cond
    ((= schrittzahl 0) 1)
    ((= schrittzahl 1) 1)
    ((> schrittzahl 1)
        (+ (fibonacci (- schrittzahl 1))
           (fibonacci (- schrittzahl 2))))
  )
)
```

Verifikation

Programmierparadigmen (Fortsetzung)

- deklarativ z.B. Prolog

```
fibonacci(0,1).
fibonacci(1,1).
fibonacci(X,Y) :- X1 is X-1, fibonacci(X1,Y1),
                  X2 is X-2, fibonacci(X2,Y2),
                  Y is X1+X2.
```

- objektorientiert z.B. Java

```
class Fibonacci extends IntegerSequence
                implements Displayable {
   int computeMember(int x) {
      ...
   }
}
```


Verifikation

Programmierparadigmen (Fortsetzung)

- deklarativ z.B. Prolog

```
fibonacci(0,1).
fibonacci(1,1).
fibonacci(X,Y) :- X1 is X-1, fibonacci(X1,Y1),
                  X2 is X-2, fibonacci(X2,Y2),
                  Y is X1+X2.
```

- objektorientiert z.B. Java

```
class Fibonacci extends IntegerSequence
                implements Displayable {
   int computeMember(int x) {
      ...
   }
}
```

Eine Verifikation ist angepasst an Programmierparadigma

→ Wir betrachten hier als Beispiel eine einfache imperative Programmiersprache.

Verifikation

Eine einfache imperative Programmiersprache

Variable

- Integer-Variable und Float-Variable

  Variable zur Aufnahme von Zahlenwerten beliebiger Größe/Genauigkeit

Konstanten

- Integer-Konstanten und Float-Konstanten

  Angabe fester Zahlenwerte

Verifikation

Eine einfache imperative Programmiersprache

Variable

- Integer-Variable und Float-Variable

  Variable zur Aufnahme von Zahlenwerten beliebiger Größe/Genauigkeit

Konstanten

- Integer-Konstanten und Float-Konstanten

  Angabe fester Zahlenwerte

Ausdrücke

- Arithmetic Expression

  Arithmetische Ausdrücke mit Zahlenwerten als Ergebnissen bestehend aus Konstanten, Variablen und Operatoren $(+, -, *, /, ...)$

- Boolean Expression

  Boolesche Ausdrücke bestehend aus arithmetische Ausdrücken mit Vergleichsoperatoren $(<, >, =, \leq, ...)$ und den Booleschen Konnektoren $(\text{and}, \text{or}, \text{not})$.

Verifikation

Eine einfache imperative Programmiersprache (Fortsetzung)

Elementare Anweisungen (Statements)

- Zuweisung

  <Variable> := <Arithmetic-Expression> ;

- Bedingte Anweisung
  - einseitig

    `if` (<Boolean-Expression>)
    `then` <Anweisung>

  - zweiseitig

    `if` (<Boolean-Expression>)
    `then` <Anweisung>
    `else` <Anweisung>

- Schleife

  `while` (<Boolean-Expression>) `do`
  <Anweisung>

Verifikation

Eine einfache imperative Programmiersprache (Fortsetzung)

Anweisungen und Programm

- Anweisungsblock

  ```
  begin
      <Anweisungsfolge>
  end
  ```

- Anweisungsfolge

  Eine Anweisungsfolge besteht aus einer endlichen Folge von Anweisungen. Eine Anweisungsfolge kann nicht leer sein.

- Anweisung

  Eine Anweisung ist eine Zuweisung, eine bedingte Anweisung, eine Schleife oder ein Anweisungsblock.

- Programm

  Ein Programm besteht aus einem Anweisungsblock.

Bemerkungen

- Zur Vereinfachung verwenden wir eine Single-Entry-Single-Exit-Sprache: Es gibt nur jeweils eine Stelle, an der bei einem Programmaufruf die Ausführung beginnt und ebenso genau eine Stelle, an der die Programmausführung endet.
- Wir geben keine Deklaration in Form eines Prozedur- oder Funktionskopfes an und verzichten auf ein explizites Return-Statement.
- Der später vorgestellte Ansatz zur Verifikation läßt sich aber in einfacher Weise auf Prozedur- und Funktionsaufrufe in Expressions erweitern. Rekursive Funktionen verlangen jedoch ein ähnlich komplexes Vorgehen wie bei den Schleifen und ausserdem sollte man temporäre Variablen und die Sichtbarkeit von Variablen einführen.

Verifikation

Beispiel für ein Programm

Programm P:

$$
\begin{array}{l}
\texttt{begin} \\
\quad a := x; \\
\quad b := y; \\
\quad \texttt{while}\ (a > 0)\ \texttt{do} \\
\qquad \texttt{begin} \\
\qquad\quad a := a - 1; \\
\qquad\quad b := b + 1; \\
\qquad \texttt{end} \\
\texttt{end}
\end{array}
$$

Was leistet dieses Programm?

Einfache imperative Programmiersprache (Fortsetzung)

Vereinbarungen

- Eingabe:

  Die Eingabewerte für ein Programm werden in speziellen Variablen übergeben. Diese Variablen werden durch das Programm *nicht* verändert.

- Initialisierung:

  Alle anderen Variablen enthalten einen beliebigen Wert, müssen also initialisiert werden. Außer den Eingabevariablen dürfen die Werte aller anderen Variablen überschrieben werden.

- Ausgabe:

  Das Ergebnis des Programmes wird in einer Variablen gespeichert, die keinen Eingabewert bereitstellt.

- Ausgabe:

  Das Ergebnis des Programmes steht in dieser Variablen auch nach dem Programmende zur Verfügung. Variablen sind also *global*.

Verifikation

Beispiel für ein Programm

Programm P:

Eingabevariable x, y
Ausgabevariable b

```
begin
    a := x;
    b := y;
    while (a > 0) do
        begin
            a := a − 1;
            b := b + 1;
        end
end
```

Was leistet dieses Programm?

Verifikation

Semantik formaler Sprachen

- Übersetzersemantik

- Operationale Semantik

- Denotationelle Semantik

- Axiomatische Semantik

Verifikation

Semantik formaler Sprachen

- **Übersetzersemantik**

  Bedeutung wird vom Übersetzer (Compiler) durch Transformation in eine (einfache) Zielsprache bestimmt.

- **Operationale Semantik**

  Bedeutung wird durch Abläufe in einer abstrakten Maschine (Automat) bestimmt.

- **Denotationelle Semantik**

  Bedeutung wird durch eine Funktion $f : E \rightarrow A$ definiert, die die Eingangszustände E auf die Ausgangszustände A abbildet.

- **Axiomatische Semantik**

  Bedeutung wird durch Axiome zur Beschreibung von Voraussetzungen und Ergebnissen von Anweisungen bestimmt.

Verifikation mit dem Hoare-Kalkül

Der Hoare-Kalkül (auch Hoare-Logik) ist ein Formales System, entwickelt von dem britischen Informatiker C.A.R. Hoare und später verfeinert von Hoare und anderen Wissenschaftlern. Er wurde 1969 in einem Artikel mit dem Titel „An Axiomatic Basis for Computer Programming“ veröffentlicht. Der Zweck des Systems ist es, eine Menge von logischen Regeln zu liefern, die es erlauben, Aussagen über die Korrektheit von imperativen Computer-Programmen zu treffen und sich dabei der mathematischen Logik zu bedienen.

[Wikipedia, 2009]

Verifikation mit dem Hoare-Kalkül

Imperative Programmierung

- Zustandsorientierte Programmierung
- Zustand $=$ aktuelle Belegung aller Variablen an einer Stelle im Programm (auf abstraktem Niveau, kein Systemstack, etc.)
- Ausführung von Anweisungen verändert Variablenbelegungen.
- Jede Anweisung kann einzeln betrachtet werden. (Anweisung $\neq$ Zeile!!!)
- Aus der Beschreibung des Zustands vor einer Anweisung läßt sich der Zustand nach der Instruktion ableiten.

$$\text{Zustand} \xrightarrow[\text{Anweisung}]{} \text{Zustand} \xrightarrow[\text{Anweisung}]{} \text{Zustand} \xrightarrow[\text{Anweisung}]{} \cdots \xrightarrow[\text{Anweisung}]{} \text{Zustand}$$

➜ Beschreibung von Zuständen durch sogenannte Zusicherungen.

➜ Zusicherungen abstrahieren so weit, dass alle Zustände erfasst sind, die an einer Stelle möglich sind (Schleifen!).

Verifikation mit dem Hoare-Kalkül

Annotation durch Zusicherungen

- *Zusicherungen sind (mathematische) Aussagen über die (Werte der) Variablen in einem Programm.*
- Zusicherungen enthalten Programmvariablen als freie Identifikatoren.
- Jede Zusicherung bezieht sich auf eine bestimmte Stelle in einem Programm.
  (Mögliche Stellen sind *nur* die Stellen vor oder nach Anweisungen.)
- Zusicherungen werden in geschweifte Klammern eingeschlossen.
  Beispiel $\{x \in \mathrm{N}$ *und* $x \geq 5$ *und* $y > 2x\}$
- Man kann Zusicherungen als (formale) Kommentare in Programmen ansehen.
- *Zusicherungen sind gültig, falls sie in jedem an der entsprechenden Stelle während eines Programmablaufes möglicherweise auftretenden Zustand erfüllt sind (Floyd/Hoare).*

Verifikation mit dem Hoare-Kalkül

Spezifikation als spezielle Zusicherungen

Die Spezifikation eines Programms P besteht aus

1. einer Zusicherung V in den Eingabevariablen des Programms, die *Vorbedingung (precondition)* von P genannt wird,
2. einer Zusicherung N in den Ausgabevariablen des Programms, die *Nachbedingung (postcondition)* von P genannt wird.

- V soll die erlaubten Eingaben für das Programm beschreiben.
- N beschreibt, welches Ergebnis für diese Eingaben in welchen Ausgabevariablen berechnet werden soll.
- Man schreibt kurz: $\{V\}P\{N\}$

➜ Während die Gültigkeit von V als gegeben angenommen wird, wollen wir uns von der Gültigkeit von N überzeugen.

Verifikation mit dem Hoare-Kalkül

Beispiel für eine Spezifikation

P sei ein Programm zur Fakultätsberechnung.

Eingabevariable n
Ausgabevariable y

Spezifikation:

Programm P:

Vorbedingung $\{ n \in \mathbf{Z}$ *und* $n \geq 0\}$
Nachbedingung $\{ y = n!$ *und* $n \in \mathbf{Z}$ *und* $n \geq 0\}$

Verifikation mit dem Hoare-Kalkül

Verifikation auf Basis von Zusicherungen

- Die Spezifikation eines Programmes beschreibt
  - in der Vorbedingung eine Abstraktion des (für das Programm relevanten Teil des) Zustands vor der Programmausführung und
  - in der Nachbedingung eine Abstraktion des (für den Programmbenutzer relevanten Teil des) Zustands nach der Programmausführung.

Verifikation mit dem Hoare-Kalkül

Verifikation auf Basis von Zusicherungen

- Die Spezifikation eines Programmes beschreibt
  - in der Vorbedingung eine Abstraktion des (für das Programm relevanten Teil des) Zustands vor der Programmausführung und
  - in der Nachbedingung eine Abstraktion des (für den Programmbenutzer relevanten Teil des) Zustands nach der Programmausführung.

- Analog zur Spezifikation des gesamten Programmes können wir Spezifikationen einzelner Anweisungen betrachten.
  - Vorbedingung einer Anweisung
    ist eine Zusicherung vor der Ausführung der Anweisung.
  - Nachbedingung einer Anweisung
    ist eine Zusicherung für den aus der Vorbedingung durch Ausführung der Anweisung resultierenden Zustand.

Verifikation mit dem Hoare-Kalkül

Verifikation auf Basis von Zusicherungen

- Die Spezifikation eines Programmes beschreibt
  - in der Vorbedingung eine Abstraktion des (für das Programm relevanten Teil des) Zustands vor der Programmausführung und
  - in der Nachbedingung eine Abstraktion des (für den Programmbenutzer relevanten Teil des) Zustands nach der Programmausführung.

- Analog zur Spezifikation des gesamten Programmes können wir Spezifikationen einzelner Anweisungen betrachten.
  - Vorbedingung einer Anweisung
    ist eine Zusicherung vor der Ausführung der Anweisung.
  - Nachbedingung einer Anweisung
    ist eine Zusicherung für den aus der Vorbedingung durch Ausführung der Anweisung resultierenden Zustand.

➜ Bei der Verifikation eines Programmes mit einer vorgegebenen Spezifikation werden die Zusicherungen vor und nach jeder Anweisung des Programmes untersucht.

Verifikation mit dem Hoare-Kalkül

Verifikation auf Basis von Zusicherungen (Fortsetzung)

- Wie verändert eine Anweisung den Zustand und damit die Zusicherung?

  ➜ Für jeden Typ von Anweisungen gibt es eine eigene Verifikationsregel.

- Welcher Nachfolgezustand ergibt sich aus einem Zustand?
  Wie sah der Vorgängerzustand eines Zustands aus?

  ➜ Jede Anweisung wird einzeln verifiziert.

- Der Nachfolgezustand der einen Anweisung ist der Vorgängerzustand der nächsten Anweisung und damit die Nachbedingung der einen die Vorbedingung der nächsten Anweisung.

  ➜ Anweisungsblöcke werden durch zusammenpassende Einzelschritte verifiziert.

➜ Verifikation eines Programmes ist Beweis der Korrektheit des Programmes unter Verwendung von akzeptierten Verifikationsregeln für die verwendete Programmiersprache.

Verifikation mit dem Hoare-Kalkül

Definition 1 (Hoare-Formel)
Eine Hoare-Formel

$$\{V\}S\{N\}$$

besteht aus Zusicherungen V und N und einer Anweisungsfolge S. V heißt auch Vorbedingung, N Nachbedingung der Anweisungsfolge S.

Semantik einer Hoare-Formel:

> Für jeden (Ausgangs-)Zustand, für den vor Ausführung der Anweisung S die Zusicherung V gilt, gilt nach der Ausführung von S für den Folgezustand die Zusicherung N.

Beachte: Die Anweisungsfolge S kann ein komplexes Programmstück, aber auch nur eine Anweisung sein!

➜ Woher kommen die benötigten Hoare-Formeln?

Verifikation mit dem Hoare-Kalkül

Verifikation auf Basis axiomatischer Semantik

Axiomatische Semantik:

> Die Semantik wird durch ein Axiom $\{V\}S\{N\}$ für jede ausführbare Anweisung S der verwendeten Programmiersprache definiert.
>
> [Hoare, 1969]

Verifikation mit dem Hoare-Kalkül

Verifikation auf Basis axiomatischer Semantik

Axiomatische Semantik:

> Die Semantik wird durch ein Axiom $\{V\}S\{N\}$ für jede ausführbare Anweisung S der verwendeten Programmiersprache definiert.
>
> [Hoare, 1969]

(Totale) Korrektheit

Der Hoare-Kalkül soll einen Nachweis liefern, dass für ein Programm P eine Spezifikation $\{V\}P\{N\}$ korrekt ist.

Erforderliche Teilbeweise:

- **Partielle Korrektheit**

  Wenn das Programm P terminiert, so transformiert P jeden Zustand, in dem V gültig ist, in einen Zustand, in dem N gültig ist.

- **Terminierung**

  Das Programm P terminiert für jeden Anfangszustand, in dem V gültig ist.

Verifikation mit dem Hoare-Kalkül

Beispiel für ein Programm

Programm P:

Vorbedingung $\{x, y \in \mathbf{N}\}$
Nachbedingung $\{b = x + y$ *und* $x, y \in \mathbf{N}\}$

begin
    $a := x;$
    $b := y;$
    while $(a > 0)$ do
        begin
            $a := a - 1;$
            $b := b + 1;$
        end
end

➜ Wie können die nötigen Nachweise für die Korrektheit geführt werden?

Verifikation mit dem Hoare-Kalkül

Definition 2 (Hoare-Regel)

Eine Hoare-Regel ist ein Schema folgender Art

$$\frac{\begin{array}{c}\text{Voraussetzung 1}\\ \vdots \\ \text{Voraussetzung n}\end{array}}{\text{Schlussfolgerung}}$$

„Voraussetzung i“ ist entweder eine Hoare-Formel oder eine Formel der Art $\{V_1\} \Rightarrow \{V_2\}$, wobei V_1 und V_2 Zusicherungen sind und V_2 eine Folgerung von V_1 ist. „Schlussfolgerung“ ist wieder eine Hoare-Formel.

Hoare-Regeln können genutzt werden, um aus gegebenen Hoare-Formeln weitere Hoare-Formeln herzuleiten.

➜ Die Verifikation eines Programmes P besteht aus der Herleitung der Hoare-Formel $\{V\}P\{N\}$ mit Hilfe von Hoare-Regeln, wobei V und N die Vor- und Nachbedingung aus der Spezifikation von P sind.

Verifikation mit dem Hoare-Kalkül

Definition 3 (Hoare-Kalkül)

Ein Hoare-Kalkül für eine (einfache) imperative Programmiersprache ist ein System von Regeln passend zu den Anweisungen der Programmiersprache, die für Programme P die Ableitung von Hoare-Formeln $\{V\}P\{N\}$ erlaubt.

Der Hoare-Kalkül kann genutzt werden, um bei gegebener Vorbedingung gültige Nachbedingungen zu ermitteln oder aber um die Vorbedingungen zu ermitteln, die für eine gewünschte Nachbedingung erforderlich ist.

Definition 4 (Herleitung im Hoare-Kalkül)

Eine Herleitung einer Hoare-Formel $\{V\}S\{N\}$ in einem Hoare-Kalkül ist eine Folge von Hoare-Formeln $\{V_1\}S_1\{N_1\}, \ldots, \{V_k\}S_k\{N_k\}$, deren letzte die herzuleitende Hoare-Formel ist, $\{V\}S\{N\} = \{V_k\}S_k\{N_k\}$ und für die jede Hoare-Formel $\{V_i\}S_i\{N_i\}$ mit Hilfe einer Regel des Kalküls unter Verwendung von nur den Hoare-Formeln $\{V_1\}S_1\{N_1\}, \ldots, \{V_{i-1}\}S_{i-1}\{N_{i-1}\}$ erzeugt wurde.

Verifikation mit dem Hoare-Kalkül

Kalkül für die einfache imperative Programmiersprache

Benötigte Hoare-Regeln:

- Regel für Zuweisungen
- Regel für bedingte Anweisungen
- Regel für Schleifen
- Regeln für Anweisungsfolgen und Anweisungsblöcke
- Abschwächungsregeln (unabhängig von konkreter Sprache)

Erzeugung von Anfangsformeln:

- Manche Regeln müssen ohne Voraussetzungen auskommen.

→ Herleitungen sind baumartige Strukturen.
Wie lassen sich Herleitungen und Programme gemeinsam darstellen?

Verifikation mit dem Hoare-Kalkül

Verifikation

- Zusicherungen betreffen Zustände vor und nach Anweisungen eines Programmes.

  ➜ Zusicherungen werden unmittelbar in das Programm integriert.

- Hoare-Regeln spiegeln die Struktur von Anweisungen wider.

  ➜ Ein Programm kann nicht Zeile für Zeile, sondern nur der Struktur entsprechend bearbeitet werden.

- Abgeleitete Hoare-Formeln lassen den Nachweis der partiellen Korrektheit oder der Terminierung zu, im allgemeinen jedoch nicht beides gleichzeitig.

  ➜ Partielle Korrektheit und Terminierung von Schleifen werden mit zwei separaten Herleitungen mit dem Hoare-Kalkül gezeigt.

- Terminierung ist Voraussetzung in jeder Hoare-Formel.

Verifikation mit dem Hoare-Kalkül

Beipiel einer Verifikation: Addition zweier natürlicher Zahlen

Spezifikation

Vorbedingung $\{x, y \in \mathbf{N}\}$
Nachbedingung $\{b = x + y$ *und* $x, y \in \mathbf{N}\}$

Programm

```
begin
   a := x;
   b := y;
   while (a > 0) do
      begin
         a := a - 1;
         b := b + 1;
      end
end
```

Verifikation mit dem Hoare-Kalkül

Beipiel einer Verifikation: Addition zweier natürlicher Zahlen (Fortsetzung)

Partielle Korrektheit

$\{x, y \in \mathbf{N}\}$

`begin`

$\{x, y \in \mathbf{N}\} \quad \Rightarrow \quad \{x, y \in \mathbf{N} \text{ und } x = x\}$

$a := x;$

$\{x, y \in \mathbf{N} \text{ und } a = x\} \quad \Rightarrow \quad \{x, y \in \mathbf{N} \text{ und } a = x \text{ und } y = y\}$

$b := y;$

$\{x, y \in \mathbf{N} \text{ und } a = x \text{ und } b = y\} \Rightarrow \{x, y \in \mathbf{N} \text{ und } a + b = x + y \text{ und } a \geq 0\}$

`while` $(a > 0)$ `do`

$\{x, y \in \mathbf{N} \text{ und } a + b = x + y \text{ und } a \geq 0 \text{ und } a > 0\}$

`begin`

$\{x, y \in \mathbf{N} \text{ und } a + b = x + y \text{ und } a \geq 0 \text{ und } a > 0\}$

$\Rightarrow \{x, y \in \mathbf{N} \text{ und } a - 1 + b + 1 = x + y \text{ und } a - 1 \geq 0\}$

$a := a - 1;$

$\{x, y \in \mathbf{N} \text{ und } a + b + 1 = x + y \text{ und } a \geq 0\}$

$b := b + 1;$

$\{x, y \in \mathbf{N} \text{ und } a + b = x + y \text{ und } a \geq 0\}$

`end`

$\{x, y \in \mathbf{N} \text{ und } a + b = x + y \text{ und } a \geq 0\}$

$\{x, y \in \mathbf{N} \text{ und } a + b = x + y \text{ und } a \geq 0 \text{ und } a \leq 0\}$

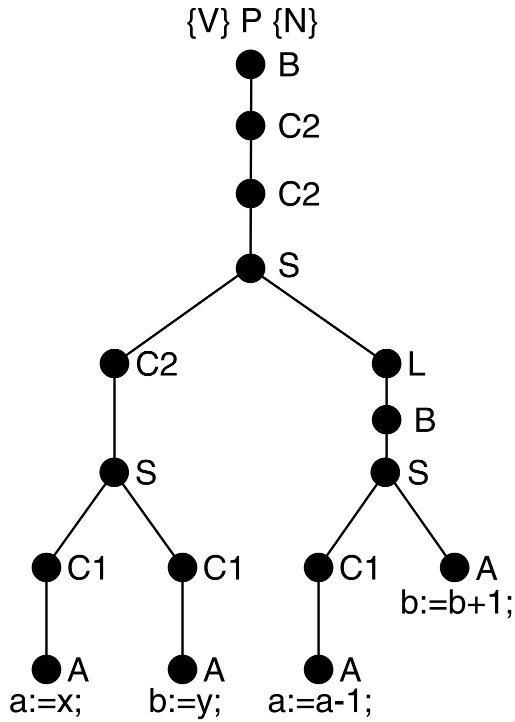
$\Rightarrow \{x, y \in \mathbf{N} \text{ und } a + b = x + y \text{ und } a = 0\} \quad \Rightarrow \quad \{x, y \in \mathbf{N} \text{ und } b = x + y\}$

`end`

$\{x, y \in \mathbf{N} \text{ und } b = x + y\}$

Verifikation mit dem Hoare-Kalkül

Herleitung im Hoare-Kalkül als Baum



- Der Herleitung, die hinter der Verifikation steht, bildet eine Baumstruktur.
- Eine Verifikation eines Programmes kann zu großen Teilen sequentiell am Programm orientiert vorgenommen werden:
  - vorwärts: Hoare-Kalkül
  - rückwärts: Weakest Preconditions nach Dijkstra

Kapitel L: V

V. Erweiterungen und Anwendungen zur Logik

- Produktionsregelsysteme
- Inferenz für Produktionsregelsysteme
- Produktionsregelsysteme mit Negation
- Nicht-monotones Schließen
- Logik und abstrakte Algebren
- Verifikation
- Verifikation mit dem Hoare-Kalkül
- Hoare-Regeln und partielle Korrektheit
- Terminierung

Hoare-Regeln und partielle Korrektheit

Hoare-Regel für Zuweisungen

$$A: \quad \frac{-}{\{N[x/e]\}\ x := e;\ \{N\}}$$

(A steht für Assignment.)

- Die Zuweisungsregel legt die Semantik der Zuweisung fest.
- Die Zuweisungsregel ordnet jeder Nachbedingung eine schwächste Vorbedingung (weakest precondition (wp)) zu.
- Die Zuweisungsregel führt die Semantik der Zuweisung auf die Semantik der Substitution in der Prädikatenlogik zurück.
- Da die Zuweisungsregel keine Voraussetzungen hat, bezeichnet man sie auch als Axiom.

Hoare-Regeln und partielle Korrektheit

Beispiele für die Anwendung der Zuweisungsregel

Abgeleitete Hoare-Formel für die Anweisung $b := y;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $y = y\}$
$b := y;$
$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

Abgeleitete Hoare-Formel für die Anweisung $x := x + 1;$

$\{x + 1 = a\}$
$x := x + 1;$
$\{x = a\}$

Hoare-Regeln und partielle Korrektheit

Abschwächungsregeln

$$C1: \quad \frac{\{P\} \;\Rightarrow\; \{P'\} \qquad \{P'\}\; S\; \{Q\}}{\{P\}\; S\; \{Q\}} \qquad\qquad C2: \quad \frac{\{P\}\; S\; \{Q'\} \qquad \{Q'\} \;\Rightarrow\; \{Q\}}{\{P\}\; S\; \{Q\}}$$

P' ist eine Abschwächung der Zusicherung P, so dass ein Zustand, der P erfüllt, auch P' erfüllt, d.h. P' ist semantische Folgerung von P, $P \models P'$.

Q ist eine Abschwächung der Zusicherung Q', so dass ein Zustand, der Q' erfüllt, auch Q erfüllt, d.h. Q ist semantische Folgerung von Q', $Q' \models Q$.

- Die Abschwächungsregeln verändern nur die Zusicherungen.
- Die Abschwächungsregeln setzen voraus, dass die Hoare-Formel für das entsprechende Programmstück bereits hergeleitet wurde.

Hoare-Regeln und partielle Korrektheit

Beispiele für die Anwendung der Abschwächungsregeln

Herleitung von Hoare-Formeln für die Anweisung $b := y;$

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{x, y \in \mathbf{N}$ und $a = x$ und $y = y\}$

$b := y;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$\Rightarrow \{x, y \in \mathbf{N}$ und $a + b = x + y$ und $a \geq 0\}$

Weitere Abschwächungen:

- Streichen konjunktiv verknüpfter Teile:

  $\{Q_1$ und $Q_2\} \;\Rightarrow\; \{Q_1\}$
- Hinzufügen von Folgerungen:

  $\{Q_1$ und $(Q_1 \Rightarrow Q_2)\} \;\Rightarrow\; \{Q_1$ und $(Q_1 \Rightarrow Q_2)$ und $Q_2\}$
- Hinzufügen von tautologischen Aussagen:

  $\{Q\} \;\Rightarrow\; \{Q$ und $x = x\}$
- Hinzufügen disjunktiv verknüpfter Teile:

  $\{Q_1\} \;\Rightarrow\; \{Q_1$ oder $Q_2\}$

Hoare-Regeln und partielle Korrektheit

Anwendung der Zuweisungsregel

$$A: \quad \frac{-}{\{N[x/e]\}\ x := e;\ \{N\}}$$

Von der Nachbedingung zur Vorbedingung (rückwärts):

- Nachbedingung N ist bekannt.
- Ersetze alle Vorkommen von x in N durch e.
- Ergebnis ist die schwächste Vorbedingung $N[x/e]$.

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel

$x := 27;$

$\{x \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$y := x;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$x := x + 1;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$y := x + y;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel

$x := 27;$

$\{x \in \mathrm{N}$ und $y \in \mathrm{N}$ und $a = x$ und $b = y\}$

$y := x;$

$\{x, y \in \mathrm{N}$ und $a = x$ und $b = y\}$

$x := x + 1;$

$\{x, y \in \mathrm{N}$ und $a = x$ und $b = y\}$

$y := x + y;$

$\{x, y \in \mathrm{N}$ und $a = x$ und $b = y\}$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel

$\{27 \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = 27$ und $b = y\}$

$x := 27;$

$\{x \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$y := x;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$x := x + 1;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$y := x + y;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

Beispiele zur Anwendung der Zuweisungsregel

$\{27 \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = 27$ und $b = y\}$

$x := 27;$

$\{x \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$y := x;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$x := x + 1;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$y := x + y;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel

$\{27 \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = 27$ und $b = y\}$

$x := 27;$

$\{x \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$\{x, x \in \mathbf{N}$ und $a = x$ und $b = x\}$

$y := x;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$x := x + 1;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$y := x + y;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel

$\{27 \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = 27$ und $b = y\}$

$x := 27;$

$\{x \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$\{x, x \in \mathbf{N}$ und $a = x$ und $b = x\}$

$y := x;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$\{x + 1, y \in \mathbf{N}$ und $a = x + 1$ und $b = y\}$

$x := x + 1;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$y := x + y;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel

$\{27 \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = 27$ und $b = y\}$

$x := 27;$

$\{x \in \mathbf{N}$ und $y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$\{x, x \in \mathbf{N}$ und $a = x$ und $b = x\}$

$y := x;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$\{x + 1, y \in \mathbf{N}$ und $a = x + 1$ und $b = y\}$

$x := x + 1;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

$\{x, x + y \in \mathbf{N}$ und $a = x$ und $b = x + y\}$

$y := x + y;$

$\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$

Hoare-Regeln und partielle Korrektheit

Anwendung der Zuweisungsregel (Fortsetzung)

$$A: \quad \frac{-}{\{N[x/e]\}\ x := e;\ \{N\}}$$

Von der Vorbedingung zur Nachbedingung (vorwärts):

Fall 1: x kommt in e nicht vor

- Vorbedingung V ist bekannt
- Schwäche Vorbedingung V ab:
  - Eliminiere *alle* Vorkommen von x in V.
  - (Erzeuge neue Vorkommen von e, z.B. $e = e$.)

  Ergebnis ist (abgeschwächte) Vorbedingung V'.
- Ersetze in V' *manche* Vorkommen von e durch x.
- Ergebnis ist die Nachbedingung N.

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$x := 27;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$x := 27;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{y \in \mathbf{N}\}$

$x := 27;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{y \in \mathbf{N}\}$

$\Rightarrow \{y \in \mathbf{N}$ und $27 \in \mathbf{N}$ und $27 = 27\}$

$x := 27;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{y \in \mathbf{N}\}$

$\Rightarrow \{y \in \mathbf{N}$ und $27 \in \mathbf{N}$ und $27 = 27\}$

$x := 27;$

$\{x, y \in \mathbf{N}$ und $x = 27\}$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{y \in \mathbf{N}\}$

$\Rightarrow \{y \in \mathbf{N}$ und $27 \in \mathbf{N}$ und $27 = 27\}$

$x := 27;$

$\{x, y \in \mathbf{N}$ und $x = 27\}$

$\{x, y \in \mathbf{N}$ und $a = x\}$

$y := x;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N} \text{ und } a = x\}$

$\Rightarrow \{y \in \mathbf{N}\}$

$\Rightarrow \{y \in \mathbf{N} \text{ und } 27 \in \mathbf{N} \text{ und } 27 = 27\}$

$x := 27;$

$\{x, y \in \mathbf{N} \text{ und } x = 27\}$

$\{x, y \in \mathbf{N} \text{ und } a = x\}$

$y := x;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{y \in \mathbf{N}\}$

$\Rightarrow \{y \in \mathbf{N}$ und $27 \in \mathbf{N}$ und $27 = 27\}$

$x := 27;$

$\{x, y \in \mathbf{N}$ und $x = 27\}$

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{x \in \mathbf{N}$ und $a = x\}$

$y := x;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{y \in \mathbf{N}\}$

$\Rightarrow \{y \in \mathbf{N}$ und $27 \in \mathbf{N}$ und $27 = 27\}$

$x := 27;$

$\{x, y \in \mathbf{N}$ und $x = 27\}$

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{x \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{x \in \mathbf{N}$ und $a = x$ und $x = x\}$

$y := x;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N} \text{ und } a = x\}$

$\Rightarrow \{y \in \mathbf{N}\}$

$\Rightarrow \{y \in \mathbf{N} \text{ und } 27 \in \mathbf{N} \text{ und } 27 = 27\}$

$x := 27;$

$\{x, y \in \mathbf{N} \text{ und } x = 27\}$

$\{x, y \in \mathbf{N} \text{ und } a = x\}$

$\Rightarrow \{x \in \mathbf{N} \text{ und } a = x\}$

$\Rightarrow \{x \in \mathbf{N} \text{ und } a = x \text{ und } x = x\}$

$y := x;$

$\{x \in \mathbf{N} \text{ und } a = x \text{ und } y = x\}$

Hoare-Regeln und partielle Korrektheit

Anwendung der Zuweisungsregel (Fortsetzung)

$$A: \quad \frac{-}{\{N[x/e]\}\ x := e;\ \{N\}}$$

Von der Vorbedingung zur Nachbedingung (vorwärts):

Fall 2: x kommt in e vor

- Vorbedingung V ist bekannt
- Schwäche Vorbedingung V ab:
  - Wandle *alle* Vorkommen von x in V in Vorkommen von e um.

  Ergebnis ist (abgeschwächte) Vorbedingung V'.
- Ersetze in V' *alle* Vorkommen von e durch x.
- Ergebnis ist die Nachbedingung N.

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$x := x + 1;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$x := x + 1;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{x + 1, y \in \mathbf{N}$ und $a + 1 = x + 1\}$

$x := x + 1;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N} \text{ und } a = x\}$

$\Rightarrow \{x + 1, y \in \mathbf{N} \text{ und } a + 1 = x + 1\}$

$x := x + 1;$

$\{x, y \in \mathbf{N} \text{ und } a + 1 = x\}$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N}$ und $a = x\}$

$\Rightarrow \{x + 1, y \in \mathbf{N}$ und $a + 1 = x + 1\}$

$x := x + 1;$

$\{x, y \in \mathbf{N}$ und $a + 1 = x\}$

$\{x, y \in \mathbf{N}$ und $a = x\}$

$y := x + y;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N} \text{ und } a = x\}$

$\Rightarrow \{x + 1, y \in \mathbf{N} \text{ und } a + 1 = x + 1\}$

$x := x + 1;$

$\{x, y \in \mathbf{N} \text{ und } a + 1 = x\}$

$\{x, y \in \mathbf{N} \text{ und } a = x\}$

$y := x + y;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathbf{N} \text{ und } a = x\}$

$\Rightarrow \{x + 1, y \in \mathbf{N} \text{ und } a + 1 = x + 1\}$

$x := x + 1;$

$\{x, y \in \mathbf{N} \text{ und } a + 1 = x\}$

$\{x, y \in \mathbf{N} \text{ und } a = x\}$

$\Rightarrow \{x, x + y \in \mathbf{N} \text{ und } a = x\}$

$y := x + y;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Zuweisungsregel (Fortsetzung)

$\{x, y \in \mathrm{N}$ und $a = x\}$

$\Rightarrow \{x + 1, y \in \mathrm{N}$ und $a + 1 = x + 1\}$

$x := x + 1;$

$\{x, y \in \mathrm{N}$ und $a + 1 = x\}$

$\{x, y \in \mathrm{N}$ und $a = x\}$

$\Rightarrow \{x, x + y \in \mathrm{N}$ und $a = x\}$

$y := x + y;$

$\{x, y \in \mathrm{N}$ und $a = x\}$

Hoare-Regeln und partielle Korrektheit

Hoare-Regeln für Anweisungsfolgen

$$S: \quad \frac{\{P\}\ S_1\ \{Q\} \qquad \{Q\}\ S_2\ \{R\}}{\{P\}\ S_1 S_2\ \{R\}} \qquad\qquad B: \quad \frac{\{P\}\ S\ \{Q\}}{\{P\}\ \texttt{begin}\ S\ \texttt{end}\ \{Q\}}$$

- Die Regeln legen fest, wie Folgen und Blöcke von Anweisungen ausgeführt werden.
- Hoare-Formeln für Anweisungsfolgen müssen zusammenpassen.
- Die Strukturierung der Programme durch *begin* und *end* sorgt für eine eindeutige Aufteilung eines Programmes in Anweisungen. Aus Sicht der Verifikation wäre sie nicht erforderlich, da die Struktur des Programmes im Verifikationsbeweis festgelegt wäre.
- Die Blockung von Anweisungen erfordert keinen zusätzlichen Verifikationsaufwand.

Hoare-Regeln und partielle Korrektheit

Vorgehen bei der Verifikation eines Programmes

- Gegeben ist eine Spezifikation mit Vorbedingung $\{V\}$ und Nachbedingung $\{N\}$ sowie ein Programm P.
- P bestehe nur aus einer Folge von Zuweisungen.

Vorgehen im Hoare-Kalkül

0. Generelle Vereinfachung:
   Die Nachbedingung einer Anweisung wird durch Abschwächung zur Vorbedingung der nächsten.
1. Ergänze $\{V\}$ als Vorbedingung vor dem Programmtext.
2. Je nach Programmanweisung verfahre folgendermaßen:
   (a) `begin`
   Kopiere die Vorbedingung dieser Zeile vor die Anweisungsfolge.
   (b) `end`
   Kopiere die Nachbedingung der Anweisungsfolge hinter diese Zeile.
   (c) Zuweisung $x := e;$
   Wende Abschwächungsregeln und Zuweisungsregel auf die Vorbedingung so an, wie es die Zuweisung erfordert
   (Fall 1: x kommt in e nicht vor; Fall 2: x kommt in e vor).
3. Wende Abschwächungsregeln auf die Nachbedingung an, um die Nachbedingung $\{N\}$ der Spezifikation zu erreichen.

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation eines Programmes

Spezifikation:

Vorbedingung: $\{x, y \in \mathbf{N}\}$
Nachbedingung: $\{b = x + y\}$

```
begin
    a := x;
    b := y;
    b := b + a;
end
```

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation eines Programmes

Spezifikation:

Vorbedingung: $\{x, y \in \mathbf{N}\}$
Nachbedingung: $\{b = x + y\}$

$\{x, y \in \mathbf{N}\}$
`begin`
$\quad a := x;$
$\quad b := y;$
$\quad b := b + a;$
`end`

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation eines Programmes

Spezifikation:

Vorbedingung: $\{x, y \in \mathbf{N}\}$
Nachbedingung: $\{b = x + y\}$

```
{x, y ∈ N}
begin
    {x, y ∈ N}

    a := x;


    b := y;


    b := b + a;

end
```

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation eines Programmes

Spezifikation:

Vorbedingung: $\{x, y \in \mathbf{N}\}$
Nachbedingung: $\{b = x + y\}$

(1) $\{x, y \in \mathbf{N}\}$
(2) `begin`
(3) $\{x, y \in \mathbf{N}\}$
(4) $\Rightarrow \{x, y \in \mathbf{N}$ und $x = x\}$
(5) $a := x;$
(6)
(7)
(8) $b := y;$
(9)
(10)
(11) $b := b + a;$
(12)
(13) `end`
(14)
(15)

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation eines Programmes

Spezifikation:

Vorbedingung: $\{x, y \in \mathbf{N}\}$
Nachbedingung: $\{b = x + y\}$

$\{x, y \in \mathbf{N}\}$

```
begin
```

$\{x, y \in \mathbf{N}\}$

$\Rightarrow \{x, y \in \mathbf{N} \text{ und } x = x\}$

$a := x;$

$\{x, y \in \mathbf{N} \text{ und } a = x\}$

$b := y;$

$b := b + a;$

```
end
```

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation eines Programmes

Spezifikation:

Vorbedingung: $\{x, y \in \mathbf{N}\}$
Nachbedingung: $\{b = x + y\}$

(1) $\{x, y \in \mathbf{N}\}$
(2) `begin`
(3) $\{x, y \in \mathbf{N}\}$
(4) $\Rightarrow \{x, y \in \mathbf{N}$ und $x = x\}$
(5) $a := x;$
(6) $\{x, y \in \mathbf{N}$ und $a = x\}$
(7) $\Rightarrow \{x, y \in \mathbf{N}$ und $a = x$ und $y = y\}$
(8) $b := y;$
(9) $\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$
(10) $\Rightarrow \{x, y \in \mathbf{N}$ und $a = x$ und $b + a = a + y\}$
(11) $b := b + a;$
(12) $\{x, y \in \mathbf{N}$ und $a = x$ und $b = a + y\}$
(13) `end`
(14) $\{x, y \in \mathbf{N}$ und $a = x$ und $b = a + y\}$
(15)

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation eines Programmes

Spezifikation:

Vorbedingung: $\{x, y \in \mathbf{N}\}$
Nachbedingung: $\{b = x + y\}$

(1) $\{x, y \in \mathbf{N}\}$
(2) `begin`
(3) $\{x, y \in \mathbf{N}\}$
(4) $\Rightarrow \{x, y \in \mathbf{N}$ und $x = x\}$
(5) $a := x;$
(6) $\{x, y \in \mathbf{N}$ und $a = x\}$
(7) $\Rightarrow \{x, y \in \mathbf{N}$ und $a = x$ und $y = y\}$
(8) $b := y;$
(9) $\{x, y \in \mathbf{N}$ und $a = x$ und $b = y\}$
(10) $\Rightarrow \{x, y \in \mathbf{N}$ und $a = x$ und $b + a = a + y\}$
(11) $b := b + a;$
(12) $\{x, y \in \mathbf{N}$ und $a = x$ und $b = a + y\}$
(13) `end`
(14) $\{x, y \in \mathbf{N}$ und $a = x$ und $b = a + y\}$
(15) $\Rightarrow \{b = x + y\}$

Bemerkungen

- Beim Aufschreiben eines Programmes wird jede Anweisung in einer neuen Zeile begonnen.
- Die Zeilen im Programm werden so eingerückt, dass die Schachtelung der Anweisungen erkennbar ist.
- Die Zusicherungen werden in das Programm eingefügt.
- Die Einrückung der Zusicherungen erfolgt so, dass sie auf gleicher Höhe mit den zugehörigen Anweisungen stehen.

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation eines Programmes (Fortsetzung)

Herleitung im Hoare-Kalkül

Nr.	Hoare-Formel	Regel
a.	(4)(5)(6)	nach Zuweisungsregel A
b.	(3)(5)(6)	nach Abschwächungsregel C1 für a
c.	(7)(8)(9)	nach Zuweisungsregel A
d.	(6)(8)(9)	nach Abschwächungsregel C1 für c
e.	(4)(5)(8)(9)	nach Sequenzenregel S für b,d
f.	(10)(11)(12)	nach Zuweisungsregel A
g.	(9)(11)(12)	nach Abschwächungsregel C1 für f
h.	(4)(5)(8)(11)(12)	nach Sequenzenregel S für e,g
i.	(2)(3)(5)(8)(11)(13)(14)	nach Sequenzenregel S für h
j.	(2)(3)(5)(8)(11)(13)(15)	nach Abschwächungsregel C2 für i

Dieser Beweis zeigt nur die partielle Korrektheit, d.h. es muss zusätzlich gezeigt werden, dass das Programm terminiert.

Bemerkungen zum Lesen der Tabelle:

- Die Elemente der Herleitung sind in Spalte 1 durchnummeriert (a-z).
- Die Liste der Hoare-Formeln wird untereinander in Spalte 2 der Tabelle angegeben.
- Die Hoare-Formeln werden durch Referenz auf die Zeilennummern in dem durch Zusicherungen ergänzten Programm angegeben ((1) - (9999)).
- Es wird in Spalte 3 die angewendete Regel angegeben und eine die Referenzen auf die Hoare-Formeln, die ihre Voraussetzung bilden.

Hoare-Regeln und partielle Korrektheit

Hoare-Regeln für bedingte Anweisungen

$$I1: \quad \frac{\{B \text{ und } P\}\ S_1\ \{Q\} \qquad \{(\text{nicht } B) \text{ und } P\} \Rightarrow \{Q\}}{\{P\}\ \texttt{if}\ (B)\ \texttt{then}\ S_1\ \{Q\}}$$

$$I2: \quad \frac{\{B \text{ und } P\}\ S_1\ \{Q\} \qquad \{(\text{nicht } B) \text{ und } P\}\ S_2\ \{Q\}}{\{P\}\ \texttt{if}\ (B)\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{Q\}}$$

- In der Regel für bedingte Anweisungen wird die Bedingung der Anweisung für zusätzliche Bedingungen in den Zusicherungen verwendet.
- Da die Handlungsfäden des Programmes wieder zusammenlaufen, muss in beiden Alternativen dieselbe Nachbedingung erreicht werden.
- Eine gemeinsame Nachbedingung kann durch Abschwächungsregeln erreicht werden:
  - durch Abstraktion wie im Beispiel oder
  - durch disjunktive Verknüpfung der einzelnen Nachbedingungen.

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Regeln für bedingte Anweisungen

$\{a \in \mathbf{Z}\}$

`if` $(a > 0)$ `then`

$b := a;$

`else`

$b := -a;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Regeln für bedingte Anweisungen

$\{a \in \mathbf{Z}\}$

`if` $(a > 0)$ `then`

$\{a \in \mathbf{Z}$ und $a > 0\}$

$b := a;$

`else`

$\{a \in \mathbf{Z}$ und (nicht $a > 0)\}$

$b := -a;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Regeln für bedingte Anweisungen

$\{a \in \mathbf{Z}\}$

`if` $(a > 0)$ `then`

$\{a \in \mathbf{Z}$ und $a > 0\}$

$\Rightarrow \{a \in \mathbf{Z}$ und $a > 0$ und $a = a\}$

$b := a;$

`else`

$\{a \in \mathbf{Z}$ und (nicht $a > 0)\}$

$\Rightarrow \{a \in \mathbf{Z}$ und $a \leq 0$ und $-a = -a\}$

$b := -a;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Regeln für bedingte Anweisungen

$\{a \in \mathbf{Z}\}$

`if` $(a > 0)$ `then`

$\{a \in \mathbf{Z}$ und $a > 0\}$

$\Rightarrow \{a \in \mathbf{Z}$ und $a > 0$ und $a = a\}$

$b := a;$

$\{a \in \mathbf{Z}$ und $a > 0$ und $b = a\}$

`else`

$\{a \in \mathbf{Z}$ und (nicht $a > 0)\}$

$\Rightarrow \{a \in \mathbf{Z}$ und $a \leq 0$ und $-a = -a\}$

$b := -a;$

$\{a \in \mathbf{Z}$ und $a \leq 0$ und $b = -a\}$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Regeln für bedingte Anweisungen

$\{a \in \mathbf{Z}\}$

$\texttt{if}\ (a > 0)\ \texttt{then}$

$\{a \in \mathbf{Z}$ und $a > 0\}$

$\Rightarrow \{a \in \mathbf{Z}$ und $a > 0$ und $a = a\}$

$b := a;$

$\{a \in \mathbf{Z}$ und $a > 0$ und $b = a\}$

$\Rightarrow \{a \in \mathbf{Z}$ und $b = |a|\}$

$\texttt{else}$

$\{a \in \mathbf{Z}$ und (nicht $a > 0$)$\}$

$\Rightarrow \{a \in \mathbf{Z}$ und $a \leq 0$ und $-a = -a\}$

$b := -a;$

$\{a \in \mathbf{Z}$ und $a \leq 0$ und $b = -a\}$

$\Rightarrow \{a \in \mathbf{Z}$ und $b = |a|\}$

$\{a \in \mathbf{Z}$ und $b = |a|\}$

Hoare-Regeln und partielle Korrektheit

Anwendung der Regeln für bedingte Anweisungen

Vorgehensweise bei bekannter Vorbedingung $\{P\}$

- Vorbedingung zu den Vorbedingungen $\{P$ und $B\}$ im *then*-Teil und $\{P$ und $($nicht $B)\}$ im *else*-Teil ergänzen.
- *then*-Teil und *else*-Teil verifizieren mit Nachbedingungen $\{Q_1\}$ bzw. $\{Q_2\}$.
- Nachbedinungen zu gemeinsamer Nachbedingung $\{Q\}$ abschwächen (z.B. $(Q_1$ oder $Q_2)$ für Q wählen).
- Nachbedingung $\{Q\}$ der bedingten Anweisung einfügen.

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Regeln für bedingte Anweisungen (Fortsetzung)

$\{a > 0$ und $b > 0$ und $a \neq b\}$

`if` $(a > b)$ `then`

$a := a - b;$

`else`

$b := b - a;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Regeln für bedingte Anweisungen (Fortsetzung)

$\{a > 0$ und $b > 0$ und $a \neq b\}$

`if` $(a > b)$ `then`

$\{a > 0$ und $b > 0$ und $a \neq b$ und $a > b\}$

$a := a - b;$

`else`

$\{a > 0$ und $b > 0$ und $a \neq b$ und $a \leq b\}$

$b := b - a;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Regeln für bedingte Anweisungen (Fortsetzung)

$\{a > 0$ und $b > 0$ und $a \neq b\}$

$\texttt{if}\ (a > b)\ \texttt{then}$

$\{a > 0$ und $b > 0$ und $a \neq b$ und $a > b\}$

$\Rightarrow \{a - b + b > 0$ und $b > 0$ und $a - b + b \neq b$ und $a - b + b > b\}$

$a := a - b;$

$\texttt{else}$

$\{a > 0$ und $b > 0$ und $a \neq b$ und $a \leq b\}$

$\Rightarrow \{a > 0$ und $b - a + a > 0$ und $a \neq b - a + a$ und $a \leq b - a + a\}$

$b := b - a;$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Regeln für bedingte Anweisungen (Fortsetzung)

$\{a > 0$ und $b > 0$ und $a \neq b\}$

`if` $(a > b)$ `then`

$\{a > 0$ und $b > 0$ und $a \neq b$ und $a > b\}$

$\Rightarrow \{a - b + b > 0$ und $b > 0$ und $a - b + b \neq b$ und $a - b + b > b\}$

$a := a - b;$

$\{a + b > 0$ und $b > 0$ und $a + b \neq b$ und $a + b > b\}$

`else`

$\{a > 0$ und $b > 0$ und $a \neq b$ und $a \leq b\}$

$\Rightarrow \{a > 0$ und $b - a + a > 0$ und $a \neq b - a + a$ und $a \leq b - a + a\}$

$b := b - a;$

$\{a > 0$ und $b + a > 0$ und $a \neq b + a$ und $a \leq b + a\}$

Hoare-Regeln und partielle Korrektheit

Beispiele zur Anwendung der Regeln für bedingte Anweisungen (Fortsetzung)

$\{a > 0 \text{ und } b > 0 \text{ und } a \neq b\}$

`if` $(a > b)$ `then`

$\{a > 0 \text{ und } b > 0 \text{ und } a \neq b \text{ und } a > b\}$

$\Rightarrow \{a - b + b > 0 \text{ und } b > 0 \text{ und } a - b + b \neq b \text{ und } a - b + b > b\}$

$a := a - b;$

$\{a + b > 0 \text{ und } b > 0 \text{ und } a + b \neq b \text{ und } a + b > b\}$

$\Rightarrow \{b > 0 \text{ und } a > 0\}$

`else`

$\{a > 0 \text{ und } b > 0 \text{ und } a \neq b \text{ und } a \leq b\}$

$\Rightarrow \{a > 0 \text{ und } b - a + a > 0 \text{ und } a \neq b - a + a \text{ und } a \leq b - a + a\}$

$b := b - a;$

$\{a > 0 \text{ und } b + a > 0 \text{ und } a \neq b + a \text{ und } a \leq b + a\}$

$\Rightarrow \{a > 0 \text{ und } a + b > 0 \text{ und } 0 \neq b \text{ und } 0 \leq b\}$

$\Rightarrow \{a > 0 \text{ und } b > 0\}$

$\{a > 0 \text{ und } b > 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen

$\{a \in \mathbf{Z}$ und $a \geq 0\}$

`if` $(a$ ungerade$)$ `then`

$a := a - 1;$

// `empty else branch`

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen

$\{a \in \mathbf{Z}$ und $a \geq 0\}$

`if` $(a$ ungerade$)$ `then`

$\{a \in \mathbf{Z}$ und $a \geq 0$ und a ungerade$\}$

$a := a - 1;$

// `empty else branch`

$\{a \in \mathbf{Z}$ und $a \geq 0$ und a gerade$\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen

$\{a \in \mathbf{Z}$ und $a \geq 0\}$

`if` $(a$ ungerade$)$ `then`

$\{a \in \mathbf{Z}$ und $a \geq 0$ und a ungerade$\}$

$\Rightarrow \{a - 1 \in \mathbf{Z}$ und $a - 1 \geq 0$ und $a - 1$ gerade$\}$

$a := a - 1;$

$\{a \in \mathbf{Z}$ und $a \geq 0$ und a gerade$\}$

$//$ `empty else branch`

$\{a \in \mathbf{Z}$ und $a \geq 0$ und a gerade$\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen

$\{a \in \mathbf{Z}$ und $a \geq 0\}$

`if` $($a ungerade$)$ `then`

$\{a \in \mathbf{Z}$ und $a \geq 0$ und a ungerade$\}$

$\Rightarrow \{a - 1 \in \mathbf{Z}$ und $a - 1 \geq 0$ und $a - 1$ gerade$\}$

$a := a - 1;$

$\{a \in \mathbf{Z}$ und $a \geq 0$ und a gerade$\}$

`// empty else branch`

$\{a \in \mathbf{Z}$ und $a \geq 0$ und a gerade$\}$

$\{a \in \mathbf{Z}$ und $a \geq 0$ und a gerade$\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0\}$

$\texttt{if}\ (a > b)\ \texttt{then}$

$a := a - b;$

$//\ \texttt{empty else branch}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0\}$

if $(a > b)$ then

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0$ und $a > b\}$

$a := a - b;$

// empty else branch

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0$ und (nicht $a > b$)$\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0\}$

`if` $(a > b)$ `then`

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0$ und $a > b\}$

$\Rightarrow \{a - b, b \in \mathbf{Z}$ und $a - b \geq 0$ und $b \geq 0\}$

$a := a - b;$

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0\}$

// `empty else branch`

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0$ und $($nicht $a > b)\}$

$\Rightarrow \{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0\}$

$\texttt{if}\ (a > b)\ \texttt{then}$

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0$ und $a > b\}$

$\Rightarrow \{a - b, b \in \mathbf{Z}$ und $a - b \geq 0$ und $b \geq 0\}$

$a := a - b;$

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0\}$

$//\ \texttt{empty else branch}$

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0$ und $($nicht $a > b)\}$

$\Rightarrow \{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0\}$

$\{a, b \in \mathbf{Z}$ und $a \geq 0$ und $b \geq 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

$\{a \neq 0\}$

`if` $(a > 0)$ `then`

$b := 1;$

`else`

$b := -1;$

$\{a \cdot b > 0$ und $a \neq 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

$\{a \neq 0\}$

`if` $(a > 0)$ `then`

$\{a \neq 0$ und $a > 0\}$

$b := 1;$

`else`

$\{a \neq 0$ und $a \leq 0\}$

$b := -1;$

$\{a \cdot b > 0$ und $a \neq 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

$\{a \neq 0\}$

`if` $(a > 0)$ `then`

$\{a \neq 0$ und $a > 0\}$

$\Rightarrow \{a > 0$ und $1 = 1\}$

$b := 1;$

$\{a > 0$ und $b = 1\}$

`else`

$\{a \neq 0$ und $a \leq 0\}$

$\Rightarrow \{a < 0$ und $-1 = -1\}$

$b := -1;$

$\{a < 0$ und $b = -1\}$

$\{a \cdot b > 0$ und $a \neq 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

$\{a \neq 0\}$

$\texttt{if}\ (a > 0)\ \texttt{then}$

$\quad \{a \neq 0 \text{ und } a > 0\}$

$\quad \Rightarrow \{a > 0 \text{ und } 1 = 1\}$

$\quad b := 1;$

$\quad \{a > 0 \text{ und } b = 1\}$

$\quad \Rightarrow \{a \cdot b > 0 \text{ und } a \neq 0\}$

$\texttt{else}$

$\quad \{a \neq 0 \text{ und } a \leq 0\}$

$\quad \Rightarrow \{a < 0 \text{ und } -1 = -1\}$

$\quad b := -1;$

$\quad \{a < 0 \text{ und } b = -1\}$

$\quad \Rightarrow \{a \cdot b > 0 \text{ und } a \neq 0\}$

$\{a \cdot b > 0 \text{ und } a \neq 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

if $(a > b)$ then
  begin
    $x := a - b;$
    $b := b - a;$
    $a := x;$
  end
else
  $a := b;$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

$\{a \cdot b \geq 0\}$

`if` $(a > b)$ `then`

`begin`

$x := a - b;$

$b := b - a;$

$a := x;$

`end`

`else`

$a := b;$

$\{a \cdot b < 0\}$

Ist die Nachbedingung gültig bei dieser Vorbedingung?

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

```
{a · b ≥ 0}
if (a > b) then
   begin
      {a · b ≥ 0 und a > b}
      x := a − b;
      b := b − a;
      a := x;
      { ??? }
   end
else
   {a · b ≥ 0 und a ≤ b}
   a := b;
   { ??? }
{a · b < 0}
```

Ist die Nachbedingung gültig bei dieser Vorbedingung?

Hoare-Regeln und partielle Korrektheit

Beispiele zur Verifikation von bedingten Anweisungen (Fortsetzung)

Then-Zweig

$\{a \cdot b \geq 0\}$

`if` $(a > b)$ `then`

`begin`

$x := a - b;$

$b := b - a;$

$a := x;$

`end`

Hoare-Regeln und partielle Korrektheit

Beispiele zur Verifikation von bedingten Anweisungen (Fortsetzung)

Then-Zweig

$\{a \cdot b \geq 0\}$

`if` $(a > b)$ `then`

`begin`

$\{a \cdot b \geq 0$ und $a > b\}$

$x := a - b;$

$b := b - a;$

$a := x;$

`end`

Hoare-Regeln und partielle Korrektheit

Beispiele zur Verifikation von bedingten Anweisungen (Fortsetzung)

Then-Zweig

$\{a \cdot b \geq 0\}$

if $(a > b)$ then

begin

$\{a \cdot b \geq 0$ und $a > b\}$

$\Rightarrow \{a \cdot b \geq 0$ und $a > b$ und $a - b = a - b\}$

$x := a - b;$

$\{a \cdot b \geq 0$ und $a > b$ und $x = a - b\}$

$\Rightarrow \{a \cdot (b - a + a) \geq 0$ und $0 > b - a$ und $-x = b - a\}$

$b := b - a;$

$\{a \cdot (b + a) \geq 0$ und $0 > b$ und $-x = b\}$

$\Rightarrow \{0 > b$ und $-x = b$ und $x = x\}$

$a := x;$

$\{0 > b$ und $-x = b$ und $a = x\}$

end

Hoare-Regeln und partielle Korrektheit

Beispiele zur Verifikation von bedingten Anweisungen (Fortsetzung)

Then-Zweig

$\{a \cdot b \geq 0\}$

`if` $(a > b)$ `then`

`begin`

$\{a \cdot b \geq 0$ und $a > b\}$

$\Rightarrow \{a \cdot b \geq 0$ und $a > b$ und $a - b = a - b\}$

$x := a - b;$

$\{a \cdot b \geq 0$ und $a > b$ und $x = a - b\}$

$\Rightarrow \{a \cdot (b - a + a) \geq 0$ und $0 > b - a$ und $-x = b - a\}$

$b := b - a;$

$\{a \cdot (b + a) \geq 0$ und $0 > b$ und $-x = b\}$

$\Rightarrow \{0 > b$ und $-x = b$ und $x = x\}$

$a := x;$

$\{0 > b$ und $-x = b$ und $a = x\}$

$\Rightarrow \{0 > b$ und $a > 0\}$

$\Rightarrow \{a \cdot b < 0\}$

`end`

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

Else-Zweig

$\{a \cdot b \geq 0\}$

`if` $(a > b)$ `then`

...

`else`

$a := b;$

$\{a \cdot b < 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

Else-Zweig

$\{a \cdot b \geq 0\}$

`if` $(a > b)$ `then`

...

`else`

$\{a \cdot b \geq 0$ und $a \leq b\}$

$a := b;$

$\{a \cdot b < 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

Else-Zweig

$\{a \cdot b \geq 0\}$

`if` $(a > b)$ `then`

...

`else`

$\{a \cdot b \geq 0$ und $a \leq b\}$

$\Rightarrow \{b = b\}$

$a := b;$

$\{a = b\}$

$\{a \cdot b < 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Verifikation von bedingten Anweisungen (Fortsetzung)

Else-Zweig

$\{a \cdot b \geq 0\}$

`if` $(a > b)$ `then`

...

`else`

$\{a \cdot b \geq 0$ und $a \leq b\}$

$\Rightarrow \{b = b\}$

$a := b;$

$\{a = b\}$

$\Rightarrow \{a \cdot b = a^2\}$

$\Rightarrow \{a \cdot b \geq 0\}$

$\not\Rightarrow \{a \cdot b < 0\}$

$\{a \cdot b < 0\}$

Nachbedingung nicht gültig.

Hoare-Regeln und partielle Korrektheit

Hoare-Regel für Schleifen

$$L: \quad \frac{\{I \text{ und } B\}\ S\ \{I\}}{\{I\}\ \texttt{while}\ (B)\ \texttt{do}\ S\ \{I \text{ und } (\text{nicht } B)\}}$$

- Die Zusicherung I heißt *Invariante* der Schleife.
- Nach Voraussetzung gilt nach dem Schleifenrumpf S die Zusicherung I, wenn $(I$ und $B)$ vor dem Schleifenrumpf galt. Wenn I vor Ausführung der Schleife gilt, dann gilt I in jedem Zustand nach einer Ausführung des Schleifenrumpfes S.
- Da I nach jeder Ausführung des Schleifenrumpfes gilt, so gilt I nach der Schleife, sofern diese terminiert.
- Die Schleife wird beendet, wenn die Schleifenbedingung nicht mehr gilt. Also ist $(I$ und nicht $B)$ in diesem Fall die Nachbedingung der Schleife.
- Wenn die Schleifenbedingung von Anfang an nicht gilt, so wird die Schleife nicht durchlaufen. Da I vor der Schleife gilt, so gilt nach der Schleife ebenfalls $(I$ und nicht $B)$.

Bemerkungen

- Die Verifikation mit der Schleifenregel zeigt NICHT das Terminieren der Schleife.
- Das Finden der Invarianten stellt in der Programmverifikation i.d.R. den schwierigsten Schritt dar.
- Die Verifikation einer Schleife entspricht einem induktiven Beweis. Die Invariante entspricht der Induktionsbehauptung.

Hoare-Regeln und partielle Korrektheit

Beispiel zur Anwendung der Schleifenregel

$\{x = a$ und $y = b$ und $x \geq 0\}$

`while` $(x > 0)$ `do`

`begin`

$x := x - 1;$

$y := y + 1;$

`end`

Hoare-Regeln und partielle Korrektheit

Beispiel zur Anwendung der Schleifenregel

$\{x = a$ und $y = b$ und $x \geq 0\}$

$\Rightarrow \{x + y = a + b$ und $x \geq 0\}$ Invariante

`while` $(x > 0)$ `do`

`begin`

$x := x - 1;$

$y := y + 1;$

`end`

Hoare-Regeln und partielle Korrektheit

Beispiel zur Anwendung der Schleifenregel

$\{x = a$ und $y = b$ und $x \geq 0\}$

$\Rightarrow \{x + y = a + b$ und $x \geq 0\}$ Invariante

`while` $(x > 0)$ `do`

$\{x + y = a + b$ und $x \geq 0$ und $x > 0\}$

`begin`

$\{x + y = a + b$ und $x \geq 0$ und $x > 0\}$

$x := x - 1;$

$y := y + 1;$

$\{x + y = a + b$ und $x \geq 0\}$

`end`

$\{x + y = a + b$ und $x \geq 0\}$

$\{x + y = a + b$ und $x \geq 0$ und (nicht $x > 0$)$\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Anwendung der Schleifenregel

$\{x = a$ und $y = b$ und $x \geq 0\}$

$\Rightarrow \{x + y = a + b$ und $x \geq 0\}$ Invariante

`while` $(x > 0)$ `do`

$\{x + y = a + b$ und $x \geq 0$ und $x > 0\}$

`begin`

$\{x + y = a + b$ und $x \geq 0$ und $x > 0\}$

$\Rightarrow \{x - 1 + y + 1 = a + b$ und $x - 1 \geq 0\}$

$x := x - 1;$

$\{x + y + 1 = a + b$ und $x \geq 0\}$

$y := y + 1;$

$\{x + y = a + b$ und $x \geq 0\}$

`end`

$\{x + y = a + b$ und $x \geq 0\}$

$\{x + y = a + b$ und $x \geq 0$ und (nicht $x > 0)\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Anwendung der Schleifenregel

$\{x = a$ und $y = b$ und $x \geq 0\}$

$\Rightarrow \{x + y = a + b$ und $x \geq 0\}$ Invariante

`while` $(x > 0)$ `do`

$\{x + y = a + b$ und $x \geq 0$ und $x > 0\}$

`begin`

$\{x + y = a + b$ und $x \geq 0$ und $x > 0\}$

$\Rightarrow \{x - 1 + y + 1 = a + b$ und $x - 1 \geq 0\}$

$x := x - 1;$

$\{x + y + 1 = a + b$ und $x \geq 0\}$

$y := y + 1;$

$\{x + y = a + b$ und $x \geq 0\}$

`end`

$\{x + y = a + b$ und $x \geq 0\}$

$\{x + y = a + b$ und $x \geq 0$ und (nicht $x > 0$)$\}$

$\Rightarrow \{x + y = a + b$ und $x = 0\}$

$\Rightarrow \{y = a + b\}$

Hoare-Regeln und partielle Korrektheit

Anwendung der Regeln für bedingte Anweisungen

- Wie entdeckt man eine Invariante?
  - Bei der Programmerstellung kann eine Invariante zur Verifikation vorgegeben und als Kommentar in den Programmtext eingefügt werden.
  - Ansonsten ist die einzige Möglichkeit,
    - die Implementation vollständig zu begreifen,
    - aus Durchläufen durch den Schleifenrumpf mit Beispielwerten ein Verständnis für die Veränderung der Zustände zu entwickeln und
    - dieses Verständnis als eine Invariante zu formulieren.
- Invarianten sind nicht eindeutig.
- Invarianten sind häufig konjunktiv verknüpften Aussagen.

Hoare-Regeln und partielle Korrektheit

Beispiel zur Anwendung der Schleifenregel: Abrollen der Schleife

```
while (x > 0) do
   begin
      x := x - 1;
      y := y + 1;
   end
```

```
if (x > 0) then
   begin
      x := x - 1;
      y := y + 1;
   end

if (x > 0) then
   begin
      x := x - 1;
      y := y + 1;
   end

if (x > 0) then
   begin
      x := x - 1;
      y := y + 1;
   end

      ...
```

Hoare-Regeln und partielle Korrektheit

Beispiel zur Anwendung der Schleifenregel: Abrollen der Schleife (Fortsetzung)

Suche nach einer Invarianten

- Betrachte Werteverlauf in der Schleife für beteiligte Variablen.
- Suche invariante Zusammenhänge zwischen Variablen.

$\{x = a$ und $y = b$ und $x \geq 0\}$

while $(x > 0)$ do
  begin
    $x := x - 1;$
    $y := y + 1;$
  end

Variablenwerte bei Test der Bedingung in `while`

	a	b	x	y
0	a	b	a	b
1	a	b	$a-1$	$b+1$
2	a	b	$a-2$	$b+2$
3	a	b	$a-3$	$b+3$
4	a	b	$a-4$	$b+4$

➜ Invariante: $\{x + y = a + b$ und $x \geq 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zur Anwendung der Schleifenregel: Abrollen der Schleife (Fortsetzung)

$\{x + y = a$ und $x \geq 0\}$ Invariante

if $(x > 0)$ then

begin

$x := x - 1;$

$y := y + 1;$

end

// empty else branch

if $(x > 0)$ then

$x := x - 1;$

$y := y + 1;$

end

...

Hoare-Regeln und partielle Korrektheit

Beispiel zur Anwendung der Schleifenregel: Abrollen der Schleife (Fortsetzung)

$\{x + y = a$ und $x \geq 0\}$ Invariante

if $(x > 0)$ then

$\{x + y = a$ und $x \geq 0$ und $x > 0\}$

begin

$\{x + y = a$ und $x \geq 0$ und $x > 0\}$

$\Rightarrow \{x - 1 + y + 1 = a$ und $x - 1 \geq 0\}$

$x := x - 1;$

$\{x + y + 1 = a$ und $x \geq 0\}$

$y := y + 1;$

$\{x + y = a$ und $x \geq 0\}$

end

$\{x + y = a$ und $x \geq 0\}$

// empty else branch

$\{x + y = a$ und $x \geq 0$ und (nicht $x > 0)\} \Rightarrow \{x + y = a$ und $x \geq 0\}$

$\{x + y = a$ und $x \geq 0\}$ Invariante

if $(x > 0)$ then

begin

$x := x - 1;$

$y := y + 1;$

end

$\{x + y = a$ und $x \geq 0\}$ Invariante

...

➜ Beachte Ähnlichkeit zur Verifikation der Schleife!

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten

Programmfragment mit Vor- und Nachbedingungen:

$\{n \in \mathbf{N} \text{ und } n \geq 0\}$

```
begin
   i := 0;
   x := 0;
   while (i < n) do
      begin
         i := i + 1;
         x := x + i;
      end
end
```

$$\{n \in \mathbf{N} \text{ und } x = \sum_{i=0}^{n} i\}$$

Ist eine Kombination aus drei der folgenden Teilausdrücke als Invariante möglich?

$$I_0 = (n \in \mathbf{N}), I_1 = (n \geq i), I_2 = (2x = i^2), I_3 = (x = \frac{i(i+1)}{2}), I_4 = (i \geq n)$$

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Verifikation des Anfangsstückes

$\{n \in \mathbf{N}$ und $n \geq 0\}$

```
begin
```

$i := 0;$

$x := 0;$

```
while (i < n) do
  begin
    i := i + 1;
    x := x + i;
  end
end
```

$\{n \in \mathbf{N}$ und $x = \sum_{i=0}^{n} i\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Verifikation des Anfangsstückes

$\{n \in \mathbf{N} \text{ und } n \geq 0\}$

```
begin
```

$\{n \in \mathbf{N} \text{ und } n \geq 0\}$

$\Rightarrow \{n \in \mathbf{N} \text{ und } n \geq 0 \text{ und } 0 = 0\}$

$i := 0;$

$\{n \in \mathbf{N} \text{ und } n \geq 0 \text{ und } i = 0\}$

$\Rightarrow \{n \in \mathbf{N} \text{ und } n \geq 0 \text{ und } i = 0 \text{ und } 0 = 0\}$

$x := 0;$

$\{n \in \mathbf{N} \text{ und } n \geq 0 \text{ und } i = 0 \text{ und } x = 0\}$

```
  while (i < n) do
    begin
      i := i + 1;
      x := x + i;
    end
end
```

$\{n \in \mathbf{N} \text{ und } x = \sum_{i=0}^{n} i\}$

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $n \geq i$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

```
while (i < n) do
   begin

      i := i + 1;

      x := x + i;

   end
end
```

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $n \geq i$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

$\Rightarrow \{n \geq i$ und ...$\}$

```
while (i < n) do
   begin
```

$\{n \in \mathbf{N}$ und $n \geq i$ und ... und $n > i\}$

$i := i + 1;$

$x := x + i;$

```
   end
end
```

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $n \geq i$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$
$\Rightarrow \{n \geq i$ und ...$\}$
`while` $(i < n)$ `do`
  `begin`
    $\{n \in \mathbf{N}$ und $n \geq i$ und ... und $n > i\}$
    $\Rightarrow \{n \in \mathbf{N}$ und $n > i$ und ...$\} \Rightarrow \{n \geq i + 1$ und ...$\}$
    $i := i + 1;$
    $\{n \in \mathbf{N}$ und $n \geq i$ und ...$\}$
    $x := x + i;$
    $\{n \in \mathbf{N}$ und $n \geq i$ und ...$\}$
  `end`
`end`

Also ist Zusicherung $n \geq i$ als Invariante geeignet.

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $2x = i^2$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

```
while (i < n) do
   begin
      i := i + 1;
      x := x + i;
   end
end
```

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $2x = i^2$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

$\Rightarrow \{n \in \mathbf{N}$ und $2x = i^2$ und ...$\}$

while $(i < n)$ do

begin

$\{n \in \mathbf{N}$ und $2x = i^2$ und ... und $n > i\}$

$i := i + 1;$

$x := x + i;$

end

end

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $2x = i^2$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

$\Rightarrow \{n \in \mathbf{N}$ und $2x = i^2$ und ...$\}$

`while` $(i < n)$ `do`

  `begin`

    $\{n \in \mathbf{N}$ und $2x = i^2$ und ... und $n > i\}$

    $\Rightarrow \{n \in \mathbf{N}$ und $2x = (i + 1 - 1)^2$ und $n \geq i + 1$ und ...$\}$

    $i := i + 1;$

    $\{n \in \mathbf{N}$ und $2x = (i - 1)^2$ und $n \geq i$ und ...$\}$

    $\Rightarrow \{n \in \mathbf{N}$ und $2(x + i - i) = (i - 1)^2$ und $n \geq i$ und ...$\}$

    $x := x + i;$

    $\{n \in \mathbf{N}$ und $2(x - i) = (i - 1)^2$ und $n \geq i$ und ...$\}$

    $\Rightarrow \{n \in \mathbf{N}$ und $2x - 2i = i^2 - 2i + 1$ und $n \geq i$ und ...$\}$

  `end`

`end`

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $2x = i^2$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$
$\Rightarrow \{n \in \mathbf{N}$ und $2x = i^2$ und ...$\}$
`while` $(i < n)$ `do`
  `begin`
    $\{n \in \mathbf{N}$ und $2x = i^2$ und ... und $n > i\}$
    $\Rightarrow \{n \in \mathbf{N}$ und $2x = (i + 1 - 1)^2$ und $n \geq i + 1$ und ...$\}$
    $i := i + 1;$
    $\{n \in \mathbf{N}$ und $2x = (i - 1)^2$ und $n \geq i$ und ...$\}$
    $\Rightarrow \{n \in \mathbf{N}$ und $2(x + i - i) = (i - 1)^2$ und $n \geq i$ und ...$\}$
    $x := x + i;$
    $\{n \in \mathbf{N}$ und $2(x - i) = (i - 1)^2$ und $n \geq i$ und ...$\}$
    $\Rightarrow \{n \in \mathbf{N}$ und $2x - 2i = i^2 - 2i + 1$ und $n \geq i$ und ...$\}$
    $\not\Rightarrow \{n \in \mathbf{N}$ und $2x = i^2$ und ...$\}$
  `end`
`end`

Also ist Zusicherung $2x = i^2$ als Invariante NICHT geeignet.

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $x = \frac{1}{2}i(i+1)$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

```
while (i < n) do
  begin
    i := i + 1;
    x := x + i;
  end
end
```

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $x = \frac{1}{2}i(i+1)$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und ...$\}$

`while` $(i < n)$ `do`

`begin`

$\{n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und ... und $n > i\}$

$i := i + 1;$

$x := x + i;$

`end`

`end`

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $x = \frac{1}{2}i(i+1)$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und ...$\}$

`while` $(i < n)$ `do`

`begin`

$\{n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und ... und $n > i\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}(i+1-1)(i+1)$ und $n \geq i+1$ und ...$\}$

$i := i + 1;$

$\{n \in \mathbf{N}$ und $x = \frac{1}{2}(i-1)i, n \geq i$ und ...$\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x + i - i = \frac{1}{2}(i-1)i$ und $n \geq i$ und ...$\}$

$x := x + i;$

$\{n \in \mathbf{N}$ und $x - i = \frac{1}{2}(i-1)i$ und $n \geq i$ und ...$\}$

`end`

`end`

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $x = \frac{1}{2}i(i+1)$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und ...$\}$

```
while (i < n) do
   begin
```

$\{n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und ... und $n > i\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}(i+1-1)(i+1)$ und $n \geq i+1$ und ...$\}$

$i := i + 1;$

$\{n \in \mathbf{N}$ und $x = \frac{1}{2}(i-1)i, n \geq i$ und ...$\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x + i - i = \frac{1}{2}(i-1)i$ und $n \geq i$ und ...$\}$

$x := x + i;$

$\{n \in \mathbf{N}$ und $x - i = \frac{1}{2}(i-1)i$ und $n \geq i$ und ...$\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}((i-1)i + 2i)$ und $n \geq i$ und ...$\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und $n \geq i$ und ...$\}$

```
   end
end
```

Also ist Zusicherung $x = \frac{1}{2}i(i+1)$ als Invariante geeignet.

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $i \geq n$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

```
      while (i < n) do
         begin
            i := i + 1;
            x := x + i;
         end
   end
```

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Test der Invarianten $i \geq n$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

$\not\Rightarrow \{n \in \mathbf{N}$ und $i \geq n$ und ...$\}$

```
while (i < n) do
   begin
      i := i + 1;
      x := x + i;
   end
end
```

Also ist Zusicherung $i \geq n$ als Invariante NICHT geeignet.

Hoare-Regeln und partielle Korrektheit

Beispiel zu Schleifeninvarianten (Fortsetzung)

Verifikation mit Invariante $\{n \in \mathbf{N}$ und $n \geq i$ und $x = \frac{1}{2}i(i+1)\}$

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

$\Rightarrow \{n \in \mathbf{N}$ und $n \geq i$ und $x = \frac{1}{2}i(i+1)\}$

`while` $(i < n)$ `do`

`begin`

$\{n \in \mathbf{N}$ und $n \geq i$ und $x = \frac{1}{2}i(i+1)$ und $n > i\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}(i+1-1)(i+1)$ und $n \geq i+1\}$

$i := i + 1;$

$\{n \in \mathbf{N}$ und $x = \frac{1}{2}(i-1)i$ und $n \geq i\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x + i - i = \frac{1}{2}(i-1)i$ und $n \geq i\}$

$x := x + i;$

$\{n \in \mathbf{N}$ und $x - i = \frac{1}{2}(i-1)i$ und $n \geq i\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}((i-1)i + 2i)$ und $n \geq i\} \Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und $n \geq i\}$

`end`

$\{n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und $n \geq i$ und $i \geq n\}$

$\Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und $n = i\} \Rightarrow \{n \in \mathbf{N}$ und $x = \frac{1}{2}n(n+1)\}$

`end`

$\{n \in \mathbf{N}$ und $x = \frac{1}{2}n(n+1)\} \Rightarrow \{n \in \mathbf{N}$ und $x = \sum_{i=0}^{n} i\}$

Damit ist die partielle Korrektheit gezeigt.

Hoare-Regeln und partielle Korrektheit

Beispiel Einfaches Potenzieren

Idee des Algorithmus: Rückführung auf die iterierte Multiplikation

Spezifikation: Vorbedingung: $\{x \in \mathbf{R} \text{ und } n \in \mathbf{N}\}$

Nachbedingung: $\{x \in \mathbf{R} \text{ und } n \in \mathbf{N} \text{ und } b = x^n\}$

```
begin
    a := x;
    b := 1;
    i := n;
    while (i > 0) do
        begin
            b := b * a;
            i := i - 1;
        end
end
```

Variable x und n sind Eingabewerte, Variable b speichert das Ergebnis.

Hoare-Regeln und partielle Korrektheit

Beispiel Einfaches Potenzieren (Fortsetzung)

Spezifikation: Vorbedingung: $\{x \in \mathbf{R}$ und $n \in \mathbf{N}\}$
Nachbedingung: $\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}\}$

```
begin
    a := x;
    b := 1;
    i := n;
    while (i > 0) do
        begin
            b := b * a;
            i := i - 1;
        end
end
```

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

Hoare-Regeln und partielle Korrektheit

Beispiel Einfaches Potenzieren (Fortsetzung)

Spezifikation: Vorbedingung: $\{x \in \mathbf{R} \text{ und } n \in \mathbf{N}\}$

Nachbedingung: $\{b = x^n\}$

$\{x \in \mathbf{R} \text{ und } n \in \mathbf{N}\}$

`begin`

$a := x;$

$b := 1;$

$i := n;$

`while` $(i > 0)$ `do`

`begin`

$b := b * a;$

$i := i - 1;$

`end`

`end`

Hoare-Regeln und partielle Korrektheit

Beispiel Einfaches Potenzieren (Fortsetzung)

Spezifikation: Vorbedingung: $\{x \in \mathbf{R} \text{ und } n \in \mathbf{N}\}$
Nachbedingung: $\{b = x^n\}$

$\{x \in \mathbf{R} \text{ und } n \in \mathbf{N}\}$

`begin`

$\{x \in \mathbf{R} \text{ und } n \in \mathbf{N}\}$

$\Rightarrow \{x \in \mathbf{R} \text{ und } n \in \mathbf{N} \text{ und } x = x\}$

$a := x;$

$\{x \in \mathbf{R} \text{ und } n \in \mathbf{N} \text{ und } a = x\}$

$\Rightarrow \{x \in \mathbf{R} \text{ und } n \in \mathbf{N} \text{ und } a = x \text{ und } 1 = 1\}$

$b := 1;$

$\{x \in \mathbf{R} \text{ und } n \in \mathbf{N} \text{ und } a = x \text{ und } b = 1\}$

$\Rightarrow \{x \in \mathbf{R} \text{ und } n \in \mathbf{N} \text{ und } a = x \text{ und } b = 1 \text{ und } n = n\}$

$i := n;$

$\{x \in \mathbf{R} \text{ und } n \in \mathbf{N} \text{ und } a = x \text{ und } b = 1 \text{ und } i = n\}$

`while` $(i > 0)$ `do`

`begin`

$b := b * a;$

$i := i - 1;$

`end`

`end`

Hoare-Regeln und partielle Korrektheit

Beispiel Einfaches Potenzieren (Fortsetzung)

Suche nach einer Invarianten

- Tautologische Aussagen sind immer invariant, aber sie helfen nicht.
- Betrachte Werteverlauf in der Schleife für beteiligte Variablen.

```
begin
    a := x; b := 1; i := n;
    while (i > 0) do
        begin
            b := b * a;
            i := i - 1;
        end
end
```

Variablenwerte bei Test der Bedingung in `while`

	x	n	a	b	i
0	x	n	x	x^0	n
1	x	n	x	x^1	$n-1$
2	x	n	x	x^2	$n-2$
3	x	n	x	x^3	$n-3$
4	x	n	x	x^4	$n-4$

Invariante I: $\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel Einfaches Potenzieren (Fortsetzung)

```
...
```

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $i = n\}$

`while` $(i > 0)$ `do`

`begin`

$b := b * a;$

$i := i - 1;$

`end`

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

Hoare-Regeln und partielle Korrektheit

Beispiel Einfaches Potenzieren (Fortsetzung)

...

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $i = n\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$ Invariante

`while` $(i > 0)$ `do`

`begin`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i > 0\}$

$b := b * a;$

$i := i - 1;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i \leq 0\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

Hoare-Regeln und partielle Korrektheit

Beispiel Einfaches Potenzieren (Fortsetzung)

...

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $i = n\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$ Invariante

`while` $(i > 0)$ `do`

`begin`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i > 0\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a * a^{i-1}$ und $i > 0\}$

$b := b * a;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^{i-1}$ und $i > 0\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^{i-1}$ und $i - 1 \geq 0\}$

$i := i - 1;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i \leq 0\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

Hoare-Regeln und partielle Korrektheit

Beispiel Einfaches Potenzieren (Fortsetzung)

...

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $i = n\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$ Invariante

`while` $(i > 0)$ `do`

`begin`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i > 0\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a * a^{i-1}$ und $i > 0\}$

$b := b * a;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^{i-1}$ und $i > 0\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^{i-1}$ und $i - 1 \geq 0\}$

$i := i - 1;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i \leq 0\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i = 0\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren

Idee des Algorithmus:

Quadrieren von Teilergebnissen spart mehr als die Hälfte der Multiplikationen.

Sei $n_k n_{k-1} \ldots n_2 n_1 n_0$ die Dualdarstellung von n, also

$$n = \sum_{i=0}^{k} n_i 2^i \qquad \text{mit } n_i = \begin{cases} 1 & \text{falls } n/2^i \text{ ungerade} \\ 0 & \text{sonst} \end{cases}$$

Dann gilt

$$x^n = x^{\sum_{i=0}^{k} n_i 2^i} = \prod_{i=0}^{k} (x^{2^i})^{n_i}$$

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

Spezifikation: Vorbedingung: $\{x \in \mathbf{R}$ und $n \in \mathbf{N}\}$
Nachbedingung: $\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

```
begin
    a := x;
    b := 1;
    i := n;
    while (i > 0) do
        begin
            if (i ungerade) then
                b := b * a;
            a := a^2;
            i := i/2;
        end
end
```

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

Spezifikation: Vorbedingung: $\{x \in \mathbf{R} \text{ und } n \in \mathbf{N}\}$
Nachbedingung: $\{x \in \mathbf{R} \text{ und } n \in \mathbf{N} \text{ und } b = x^n\}$

$\{x \in \mathbf{R} \text{ und } n \in \mathbf{N}\}$

`begin`

$a := x;$

$b := 1;$

$i := n;$

$\ldots$

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

Spezifikation: Vorbedingung: $\{x \in \mathbf{R}$ und $n \in \mathbf{N}\}$
Nachbedingung: $\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}\}$
`begin`
$\{x \in \mathbf{R}$ und $n \in \mathbf{N}\}$
$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x = x\}$
$a := x;$
$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x\}$
$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $1 = 1\}$
$b := 1;$
$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1\}$
$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $n = n\}$
$i := n;$
$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $i = n\}$
$\ldots$

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

Suche nach einer Invarianten

- Betrachte Werteverlauf in der Schleife für beteiligte Variablen.
- $n_k n_{k-1} \ldots n_2 n_1 n_0$ sei Dualzahldarstellung von n.

$a := x;\ b := 1;\ i := n;$
`while` $(i > 0)$ `do`
  `begin`
    `if` $(i$ ungerade$)$ `then`
      $b := b * a;$
    $a := a^2;$
    $i := i/2;$
  `end`

Variablenwerte bei Test der Bedingung in `while`

	x	n	a	b	i
0	x	n	x^1	1	$n_k n_{k-1} \ldots n_2 n_1 n_0$
1	x	n	x^2	x^{n_0}	$n_k n_{k-1} \ldots n_2 n_1$
2	x	n	x^4	$x^{n_1 n_0}$	$n_k n_{k-1} \ldots n_2$
3	x	n	x^8	$x^{n_2 n_1 n_0}$	$n_k n_{k-1} \ldots n_3$
4	x	n	x^{16}	$x^{n_3 n_2 n_1 n_0}$	$n_k n_{k-1} \ldots n_4$

Invariante: $\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

...

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $i = n\}$

```
while (i > 0) do
  begin
    if (i ungerade) then
      b := b * a;
    a := a^2;
    i := i/2;
  end
```

...

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

...

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $i = n\} \Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$ Inv.

`while` $(i > 0)$ `do`

`begin`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i > 0\}$

`if` $(i$ ungerade$)$ `then`

$b := b * a;$

$a := a^2;$

$i := i/2;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i \leq 0\}$

...

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

...

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $i = n\} \Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$ Inv.

`while` $(i > 0)$ `do`

`begin`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i > 0\} \Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0\}$

`if` $(i$ ungerade$)$ `then`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0$ und i ungerade $\}$

$b := b * a;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0$ und i gerade $\}$

$a := a^2;$

$i := i/2;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i \leq 0\}$

...

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

...

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $i = n\} \Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$ Inv.

`while` $(i > 0)$ `do`

`begin`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i > 0\} \Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0\}$

`if` $(i$ ungerade$)$ `then`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0$ und i ungerade $\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a * (a^2)^{[i/2]}$ und $i > 0$ und i ungerade $\}$

$b := b * a;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * (a^2)^{[i/2]}$ und $i > 0$ und i ungerade $\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * (a^2)^{[i/2]}$ und $i > 0\}$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0$ und i gerade $\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * (a^2)^{[i/2]}$ und $i > 0$ und i gerade $\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * (a^2)^{[i/2]}$ und $i > 0\}$

$a := a^2;$

$i := i/2;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i \leq 0\}$

...

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

...

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $a = x$ und $b = 1$ und $i = n\} \Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$ Inv.

`while` $(i > 0)$ `do`

`begin`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i > 0\} \Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0\}$

`if` $(i$ ungerade$)$ `then`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0$ und i ungerade $\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a * (a^2)^{[i/2]}$ und $i > 0$ und i ungerade $\}$

$b := b * a;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * (a^2)^{[i/2]}$ und $i > 0$ und i ungerade $\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * (a^2)^{[i/2]}$ und $i > 0\}$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i > 0$ und i gerade $\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * (a^2)^{[i/2]}$ und $i > 0$ und i gerade $\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * (a^2)^{[i/2]}$ und $i > 0\}$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * (a^2)^{[i/2]}$ und $i > 0\}$

$a := a^2;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^{[i/2]}$ und $i > 0\} \Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^{[i/2]}$ und $[i/2] \geq 0\}$

$i := i/2;$

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i \leq 0\}$

...

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

...

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i \leq 0\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

Hoare-Regeln und partielle Korrektheit

Beispiel Effizientes Potenzieren (Fortsetzung)

...

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i \geq 0$ und $i \leq 0\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b * a^i$ und $i = 0\}$

$\Rightarrow \{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $x^n = b\}$

`end`

$\{x \in \mathbf{R}$ und $n \in \mathbf{N}$ und $b = x^n\}$

Bemerkungen als Wiederholung

- Schleifeninvarianten sind Zusicherungen, d.h. sie beschreiben Zusammenhänge von Programmgrößen.
- Eine Invariante muss vor der Schleife gültig sein.
- Eine Invariante muss nach Durchlauf durch den Schleifenrumpf gültig sein, wenn sie vor dem Schleifenrumpf gültig war (zusammen mit der Schleifenbedingung).
- Invarianten sind zum Zeitpunkt der Implementierung leichter zu bestimmen.

Hoare-Regeln und partielle Korrektheit

Möglichkeiten zum Einsparen von Schreibarbeit

- Festlegungen des Grundbereiches wie $x \in \mathbf{R}$ oder $n \in \mathbf{N}$ werden in der Spezifikation für die Eingabevariablen benötigt. Da nach Vereinbarung die Eingabevariablen im Programm nicht verändert werden, können diese Festlegungen in alle Zusicherungen aufgenommen werden, ohne deren Gültigkeit zu verändern. Da sie aber nur sehr selten wirklich benötigt werden, kann man auf sie aus Gründen der Übersichtlichkeit verzichten.

  ➜ Festlegungen der Grundbereiche für Eingabevariablen setzen wir in den Zusicherungen als bekannt voraus.

- Alle einzelnen Bedingungen in den Zusicherungen sind konjunktiv verknüpft. Nur selten muss man Disjunktionen oder Negationen verwenden.

  ➜ Als Kurzform für die konjunktive Verknüpfung von Bedingungen wird ein Komma verwendet.

  Beispiel: $\{x^n = b * (a^2)^{[i/2]}, i > 0\}$

Kapitel L: V

V. Erweiterungen und Anwendungen zur Logik

- Produktionsregelsysteme
- Inferenz für Produktionsregelsysteme
- Produktionsregelsysteme mit Negation

- Nicht-monotones Schließen

- Logik und abstrakte Algebren

- Verifikation
- Verifikation mit dem Hoare-Kalkül
- Hoare-Regeln und partielle Korrektheit
- Terminierung

Terminierung

Der Nachweis der totalen Korrekheit eines Programms erfordert neben der bisher betrachteten partiellen Korrektheit den Nachweis seiner Terminierung des Programmes.

Wann ist Terminierung ein Problem?

- In Zuweisungen, falls der arithmetische Ausdruck nicht berechenbar ist (z.B. Division durch 0, nicht initialisierte Variable),
- in bedingten Anweisungen, falls die Bedingung nicht entschieden werden kann oder falls die Anweisungen im *then*-Teil oder im *else*-Teil nicht terminieren,
- in Anweisungsfolgen, falls eine Anweisung darin nicht terminiert,
- aber vor allem in Schleifen.

➜ Wir beschränken uns auf Terminierungsbeweise für Schleifen.

Bemerkungen

- Ausser in Schleifen kann die Terminierung garantiert werden, indem man die Art der arithmetischen oder booleschen Ausdrücke auf einfache Formen beschränkt (Addition, Subtraktion von 1, also $x := x + 1$; und Test auf 0, also $\texttt{if}\ (x = 0) \ldots$).

Terminierung

Terminierung von Schleifen

Eine Schleife

$$\texttt{while}\ (B)\ \texttt{do}\ S$$

terminiert unter der Vorbedingung P genau dann, wenn jede Ausführung von S terminiert und wenn eine Invariante I_T der Schleife existiert und ein ganzzahliger arithmetischer Ausdruck T existiert, so dass folgende Aussagen gelten:

1. $\{P\} \Rightarrow \{I\}$ ist gültige Abschwächung.
2. $\{T \leq 0$ und $I_T\} \Rightarrow \{$nicht $B\}$ ist gültige Abschwächung.
3. $\{T = t + 1$ und I_T und $B\}\ S\ \{T = t$ und $I_T\}$ ist gültige Hoare-Formel.

t ist ein Bezeichner für eine ganzzahlige Variable, die weder in T noch in der Schleife vorkommt, d.h. nicht in B und nicht in S.

T heißt auch Terminierungsfunktion oder *Variante* der Schleife.

Terminierung

Beispiel für den Nachweis der Terminierung einer Schleife

Schleife:

$\{x = a$ und $y = b$ und $x \geq 0\}$

```
while (x > 0) do
   begin
      x := x - 1;
      y := y + 1;
   end
```

Vorbedingung P: $(x + y = a + b$ und $x \geq 0)$
Invariante I_T: $(x \geq 0)$
Schleifenbedingung B: $(x > 0)$
Ganzzahliger Ausdruck T: x

Nachweis: $\{P\} \quad \Rightarrow \quad \{I_T\}$ ✓

Nachweis: $\{T \leq 0$ und $I_T\} \quad \Rightarrow \quad \{$nicht $B\}$ ✓

Bemerkungen

- Für eine Schleife

```
while (x > 25) do
   begin
      x := x - 1;
      y := y + 1;
   end
```

  wählt man als ganzzahligen Ausdruck T: $x - 25$

  Wichtig ist die Existenz einer unteren Schranke für T und das streng monotone Fallen des Wertes von T mit jedem Schleifendurchlauf.

- Als Invariante für die Terminierung kann häufig die Invariante (oder ein Teil hiervon) aus dem Nachweis der partiellen Korrektheit gewählt werden. Mit ihr läßt sich meist sofort die Beschränktheit der Werte der Variante T nachweisen.

Terminierung

Beispiel für den Nachweis der Terminierung einer Schleife (Fortsetzung)

Nachweis der Invarianz von I_T und der Dekrementierung von T

$\{x = a$ und $y = b$ und $x \geq 0\}$

`while` $(x > 0)$ `do`

`begin`

$x := x - 1;$

$y := y + 1;$

`end`

Terminierung

Beispiel für den Nachweis der Terminierung einer Schleife (Fortsetzung)

Nachweis der Invarianz von I_T und der Dekrementierung von T

$\{x = a$ und $y = b$ und $x \geq 0\}$

$\Rightarrow \{x \geq 0\}$

`while` $(x > 0)$ `do`

$\{x = i + 1$ und $x \geq 0$ und $x > 0\}$

`begin`

$\{x = t + 1$ und $x \geq 0$ und $x > 0\}$

$x := x - 1;$

$y := y + 1;$

$\{x = t$ und $x \geq 0\}$

`end`

$\{x = t$ und $x \geq 0\}$

Terminierung

Beispiel für den Nachweis der Terminierung einer Schleife (Fortsetzung)

Nachweis der Invarianz von I_T und der Dekrementierung von T

$\{x = a$ und $y = b$ und $x \geq 0\}$

$\Rightarrow \{x \geq 0\}$

`while` $(x > 0)$ `do`

$\{x = i + 1$ und $x \geq 0$ und $x > 0\}$

`begin`

$\{x = t + 1$ und $x \geq 0$ und $x > 0\}$

$\Rightarrow \{x - 1 = t$ und $x - 1 \geq 0\}$

$x := x - 1;$

$\{x = t$ und $x \geq 0\}$

$y := y + 1;$

$\{x = t$ und $x \geq 0\}$

`end`

$\{x = t$ und $x \geq 0\}$

Terminierung

Nachweis der Terminierung von Schleifen

- Insgesamt sind fünf Nachweise erforderlich:
  1. (Terminierung des Schleifenrumpfes)
  2. Folgerbarkeit dieser Invarianten I_T aus der Vorbedingung der Schleife
  3. Ganzzahligkeit von T
  4. Folgerbarkeit der Nicht-Gültigkeit der Schleifenbedingung aus $T \leq 0$ und einer Invarianten I_T
  5. Nachweis der Invarianz von I_T und Nachweis der Dekrementierung von T

- Die Invariante I_T muss nicht mit der Invarianten für den Nachweis der partiellen Korrektheit der Schleife übereinstimmen, häufig sind sie aber sehr ähnlich.

Beachte:

Die Gestalt der Hoare-Regel für die Schleife erlaubt keinen gleichzeitigen Nachweis von partieller Korrektheit und Terminierung.

Terminierung

Beispiel für den Nachweis der Terminierung einer Schleife

Gegeben sei das folgende Programm mit den angegebenen Vor- und Nachbedingungen:

$\{n \in \mathbf{N}\}$

```
begin
    i := 0;
    x := 0;
    {n ∈ N und n ≥ 0 und i = 0 und x = 0}
    while (i < n) do
        begin
            i := i + 1;
            x := x + i;
        end
end
{n ∈ N und x = Σ_{i=0}^{n} i}
```

Prüfen Sie, ob die Schleife terminiert.

Terminierung

Beispiel für den Nachweis der Terminierung einer Schleife (Fortsetzung)

Kandidaten für Variante T:
i, x oder arithmetische Ausdrücke hiermit, da nur sie in der Schleife verändert werden.

Kandidat $T = n - i$:

- Nachweis der Ganzzahligkeit: Anfangswerte und Veränderungen

  T ganzzahlig, da mit n initialisiert durch Vorbedingung und $i := 0;$ und T nur durch $i := i + 1;$ in der Schleife verändert.

- Nachweis der Abschwächung von $\{T \leq 0$ und $I_T\}$ zu $\{$nicht $B\}$:

  $\{n - i \leq 0\} \Rightarrow \{$nicht $i < n\}$ ✓

  Eine Invariante wird hier also gar nicht benötigt.

  Wir verwenden dennoch die Invariante $\{n \in \mathbf{N}$ und $n \geq i$ und $x = \frac{1}{2}i(i+1)\}$ aus dem Nachweis der partiellen Korrektheit. Dadurch kann der Nachweis für die partielle Korrektheit für den Nachweis der Terminierung recycled werden. Das erleichtert die Arbeit!

Terminierung

Beispiel für den Nachweis der Terminierung einer Schleife (Fortsetzung)

- Verifikation von S mit Vorbedingung $\{T = t + 1$ und I_T und $B\}$ zur Nachbedingung $\{T = t$ und $I_T\}$
  (t bezeichnet eine Variable, die nicht im Programm vorkommt)

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

```
while (i < n) do
  begin
     i := i + 1;
     x := x + i;
  end
```

Terminierung

Beispiel für den Nachweis der Terminierung einer Schleife (Fortsetzung)

- Verifikation von S mit Vorbedingung $\{T = t + 1$ und I_T und $B\}$ zur Nachbedingung $\{T = t$ und $I_T\}$
  (t bezeichnet eine Variable, die nicht im Programm vorkommt)

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

$\Rightarrow \{n \in \mathbf{N}$ und $n \geq i$ und $x = \frac{1}{2}i(i+1)\}$

`while` $(i < n)$ `do`

  `begin`

  $\{n - i = t + 1$ und $n \in \mathbf{N}$ und $n \geq i$ und $x = \frac{1}{2}i(i+1)$ und $i < n\}$

  $i := i + 1;$

  $x := x + i;$

  `end`

Terminierung

Beispiel für den Nachweis der Terminierung einer Schleife (Fortsetzung)

- Verifikation von S mit Vorbedingung $\{T = t + 1$ und I_T und $B\}$ zur Nachbedingung $\{T = t$ und $I_T\}$
  (t bezeichnet eine Variable, die nicht im Programm vorkommt)

$\{n \in \mathbf{N}$ und $n \geq 0$ und $i = 0$ und $x = 0\}$

$\Rightarrow \{n \in \mathbf{N}$ und $n \geq i$ und $x = \frac{1}{2}i(i+1)\}$

`while` $(i < n)$ `do`

`begin`

$\{n - i = t + 1$ und $n \in \mathbf{N}$ und $n \geq i$ und $x = \frac{1}{2}i(i+1)$ und $i < n\}$

$\Rightarrow \{n - (i+1) = t$ und $n \in \mathbf{N}$ und $x = \frac{1}{2}(i+1-1)(i+1)$ und $i + 1 \leq n\}$

$i := i + 1;$

$\{n - i = t$ und $n \in \mathbf{N}$ und $x = \frac{1}{2}(i-1)(i)$ und $i \leq n\}$

$\Rightarrow \{n - i = t$ und $n \in \mathbf{N}$ und $x + i = \frac{1}{2}(i+1)(i)$ und $i \leq n\}$

$x := x + i;$

$\{n - i = t$ und $n \in \mathbf{N}$ und $x = \frac{1}{2}i(i+1)$ und $i \leq n\}$

`end`

Damit folgt insgesamt, dass die Schleife terminiert.

Terminierung

Alternativer Nachweis der Terminierung von Schleifen

Auch dieser Nachweis verwendet einen Ausdruck T über den Variablen des Programms als Variante.

- Insgesamt sind vier Nachweise erforderlich:
  1. (Terminierung des Schleifenrumpfes)
  2. Ganzzahligkeit von T
  3. Nachweis der Dekrementierung (bzw. Inkrementierung) von T in jedem Schleifendurchlauf
  4. Nachweis von Schranken für T (d.h. $\{-\text{bound} \leq T \leq \text{bound}\}$ ist invariant).

➜ Sind die beiden Formulierungen gleichwertig?

Bemerkungen

- Für in den Werten streng monoton fallende Ausdrücke T benötigt man natürlich nur eine untere Schranke, für in den Werten streng monoton wachsende Ausdrücke T eine oberen Schranke.
- Dieser Nachweis wird im Teil der Inkrementierung von T weniger streng formal durchgeführt und ebenso im Bereich der Folgerbarkeit der Ungültigkeit der Schleifenbedingung bei Überschreiten der Schranke.

Terminierung

Beispiel für den alternativen Nachweis der Terminierung einer Schleife

Gegeben sei das folgende Programm mit den angegebenen Vor- und Nachbedingungen:

$\{n \in \mathbf{N}\}$

```
begin
    i := 0;
    x := 0;
    {n ∈ N und n ≥ 0 und i = 0 und x = 0}
    while (i < n) do
        begin
            i := i + 1;
            x := x + i;
        end
end
```

$\{n \in \mathbf{N} \text{ und } x = \sum_{i=0}^{n} i\}$

Prüfen Sie, ob die Schleife terminiert.

Terminierung

Beispiel für den alternativen Nachweis der Terminierung einer Schleife

(Fortsetzung)

Kandidat für die Variante: $T = i$

- T ist ganzzahlig, da i mit 0 in $i := 0;$ initialisiert und nur durch $i := i + 1;$ in der Schleife verändert wird.

- T wächst mit jedem Durchlauf durch den Schleifenrumpf um den Wert 1, da i im Schleifenrumpf nur durch $i := i + 1;$ verändert wird.

- (T ist nach unten beschränkt wegen Zusicherung $i = 0$ vor und Inkrementierung in der Schleife.)

  T ist nach oben beschränkt, da $n \geq i$ Teil der Invariante aus dem Nachweis der partiellen Korrektheit ist, also aus $n \geq i$ und $n > i$ (Schleifenbedingung) als Vorbedingung im Schleifenrumpf wieder $n \geq i$ als Nachbedingung des Schleifenrumpfes folgt und n unverändert bleibt.

➜ Damit folgt insgesamt, dass die Schleife terminiert.

Bemerkungen

- T muss hier nicht so eingeschränkt gewählt werden. Durch Subtraktion der Variante vom Maximalwert bei monoton wachsenden Ausdrücken oder des Minimalwertes von der Variante bei monoton fallenden Ausdrücken kann man eine Variante mit den alten Anforderungen herstellen.
- Anstelle einer formalen Herleitung wird hier durch *Hinschauen* die kontinuierliche Veränderung von T nachgewiesen.
- Auf einen Nachweis, dass bei Erreichen der Schranken durch T die Schleifenbedingung nicht mehr erfüllt wird, verzichtet man ganz. Dies schließt man aus strenger Monotonie und Beschränktheit der Variante, ohne es explizit zu erwähnen.
- Für den Nachweis der Beschränktheit von T nutzt man intensiv die Invariante aus dem Beweis der partiellen Korrektheit und argumentiert mit offensichtlichen Eigenschaften.
- Grundsätzlich ist dieser alternative Nachweis also unter Verwendung der Invarianten der partiellen Korrektheit ein vollständiger Nachweis der Terminierung. Er ist weniger formal, da auf exakte Begründungen verzichtet wird und Überzeugung durch Augenschein an die Stelle tritt.

Terminierung

Terminierung: Ein Problem?

- Manche Schleifen terminieren immer!

$\{a > 0$ und $b > 0\}$

while $(a \neq b)$ do
  begin
    while $(a > b)$ do
      $a := a - b;$
    while $(b > a)$ do
      $b := b - a;$
  end

Terminierung

Terminierung: Ein Problem?

- Manche Schleifen terminieren immer!

$\{a > 0$ und $b > 0\}$

```
while (a ≠ b) do
   begin
      while (a > b) do
         a := a − b;
      while (b > a) do
         b := b − a;
   end
```

- Manche Schleifen terminieren nicht immer!

$\{a > 0$ und $b > 0\}$

```
while (a ≠ b) do
   begin
      while (a ≥ b) do
         a := a − b;
      while (b > a) do
         b := b − a;
   end
```

Terminierung

Terminierung: Ein Problem? (Fortsetzung)

- Für manche Schleifen ist nicht bekannt, ob sie terminieren!

$\{n \in \mathbf{N}$ und $n > 0$ und $x = n\}$

$\texttt{while}\ (x \neq 1)\ \texttt{do}$
$\quad \texttt{if}\ (x$ gerade$)\ \texttt{then}$
$\quad\quad x := x/2;$
$\quad \texttt{else}$
$\quad\quad x := 3 * x - 1;$

(Collatz-Problem)

➜ Die Frage nach der Terminierung von Schleifen ist unentscheidbar.

Bemerkungen

- Das Collatz-Problem betrifft Zahlenfolgen, die nach einem einfachen Bildungsgesetz konstruiert werden:

  1. Beginne mit irgendeiner natürlichen Zahl $n > 0$.
  2. Ist n gerade, so nimm als nächste Zahl $n/2$.
  3. Ist n ungerade, so nimm als nächste Zahl $3n + 1$.

  Die so erhaltenen Zahlenfolgen endeten bei allen getesteten Auswahlen für n in dem Zyklus 4,2,1.

  Die Collatz-Vermutung lautet daher:

  > Jede so konstruierte Zahlenfolge endet im Zykel 4,2,1 egal, mit welcher positiven natürlichen Zahl man beginnt.

  Aufgrund des Bildungsgesetzes für die Zahlenfolge nennt man diese Vermutung auch die (3n + 1)-Vermutung. Die im Schleifenrumpf berechnete Funktion in x heißt Ulam's Funktion. [Wikipedia, 2009]

- Für den Nachweis der Unentscheidbarkeit kann man die Arbeitsschritte einer universellen Turingmaschine im Schleifenrumpf beschreiben und in der Schleifenbedingung überprüfen, ob die Rechnung der Turingmaschine terminiert (Halteproblem).

Terminierung

Nicht-Terminierung von Schleifen

Eine Schleife

$$\texttt{while}\ (B)\ \texttt{do}\ S$$

terminiert nicht, wenn eine Zusicherung I_{NT} existiert, so dass folgende Aussagen gelten:

1. Es gibt Eingaben, so dass $\{B$ und $I_{NT}\}$ vor der Schleife gültig ist.
2. $\{B$ und $I_{NT}\}$ ist Invariante der Schleife.

I_{NT} bezeichnet eine nur in bestimmten Eingabesituationen gültige Zusicherung.

Beim Nachweis der partiellen Korrektheit und der Terminierung müssen die verwendeten Zusicherungen in allen denkbaren Zuständen des Programmes an den entsprechenden Stellen gelten.

Bemerkungen

- Durch Nachweise für Nicht-Terminierung von Schleifen kann man Hinweise für notwendige Änderungen der Spezifikation erlangen.
- Durch Nachweise für Nicht-Terminierung von Schleifen zeigt man die Anwesenheit eines Fehlers, aber nicht die Abwesenheit!

Terminierung

Beispiel Nicht-Terminierung von Schleifen

Vorbedingung P: $\{x < y$ und $x \geq 0\}$

Gesucht ist verschärfte Vorbedingung R mit $\{R\} \Rightarrow \{P\}$

(Verschärfte Vorbedingung repräsentiert spezielle Wertebelegung für x und y.)

$\{x < y$ und $x \geq 0\}$

`while` $(x < y)$ `do`

$x := x * x + 5 * x;$

Terminierung

Beispiel Nicht-Terminierung von Schleifen

Vorbedingung P: $\{x < y$ und $x \geq 0\}$

Gesucht ist verschärfte Vorbedingung R mit $\{R\} \Rightarrow \{P\}$

(Verschärfte Vorbedingung repräsentiert spezielle Wertebelegung für x und y.)

$\{x < y$ und $x \geq 0\}$

$\{x < y$ und $x = 0\} = \{R\}$

`while` $(x < y)$ `do`

$x := x * x + 5 * x;$

(Genaugenommen muss $\{R\}$ eine Verschärfung der Spezifikation des Programmes sein.)

Terminierung

Beispiel Nicht-Terminierung von Schleifen

Vorbedingung P: $\{x < y$ und $x \geq 0\}$

Gesucht ist verschärfte Vorbedingung R mit $\{R\} \Rightarrow \{P\}$

(Verschärfte Vorbedingung repräsentiert spezielle Wertebelegung für x und y.)

Variante $I_{NT} = R$

$\{x < y$ und $x \geq 0\}$

$\{x < y$ und $x = 0\} = \{R\}$

$\Rightarrow \{x < y$ und $x = 0$ und $x < y\} = \{R$ und $B\}$

`while` $(x < y)$ `do`

$x := x * x + 5 * x;$

(Genaugenommen muss $\{R\}$ eine Verschärfung der Spezifikation des Programmes sein.)

Terminierung

Beispiel Nicht-Terminierung von Schleifen

Vorbedingung P: $\{x < y$ und $x \geq 0\}$

Gesucht ist verschärfte Vorbedingung R mit $\{R\} \Rightarrow \{P\}$

(Verschärfte Vorbedingung repräsentiert spezielle Wertebelegung für x und y.)

Variante $I_{NT} = R$

$\{x < y$ und $x \geq 0\}$

$\{x < y$ und $x = 0\} = \{R\}$

$\Rightarrow \{x < y$ und $x = 0$ und $x < y\} = \{R$ und $B\}$

`while` $(x < y)$ `do`

$\{x = 0$ und $x < y\} = \{R$ und $B\}$

$x := x * x + 5 * x;$

$\{x = 0$ und $x < y\} = \{R$ und $B\}$

(Genaugenommen muss $\{R\}$ eine Verschärfung der Spezifikation des Programmes sein.)

Terminierung

Beispiel Nicht-Terminierung von Schleifen

Vorbedingung P: $\{x < y$ und $x \geq 0\}$

Gesucht ist verschärfte Vorbedingung R mit $\{R\} \Rightarrow \{P\}$

(Verschärfte Vorbedingung repräsentiert spezielle Wertebelegung für x und y.)

Variante $I_{NT} = R$

$\{x < y$ und $x \geq 0\}$

$\{x < y$ und $x = 0\} = \{R\}$

$\Rightarrow \{x < y$ und $x = 0$ und $x < y\} = \{R$ und $B\}$

`while` $(x < y)$ `do`

$\{x = 0$ und $x < y\} = \{R$ und $B\}$

$\Rightarrow \{x * x + 5 * x = 0$ und $x * x + 5 * x < y\}$

$x := x * x + 5 * x;$

$\{x = 0$ und $x < y\} = \{R$ und $B\}$

(Genaugenommen muss $\{R\}$ eine Verschärfung der Spezifikation des Programmes sein.)

Terminierung

Beispiel Nicht-Terminierung von Schleifen

Vorbedingung P: $\{x < y$ und $x \geq 0\}$

Gesucht ist verschärfte Vorbedingung R mit $\{R\} \Rightarrow \{P\}$

(Verschärfte Vorbedingung repräsentiert spezielle Wertebelegung für x und y.)

Variante $I_{NT} = R$

$\{x < y$ und $x \geq 0\}$

$\{x < y$ und $x = 0\} = \{R\}$

$\Rightarrow \{x < y$ und $x = 0$ und $x < y\} = \{R$ und $B\}$

`while` $(x < y)$ `do`

  $\{x = 0$ und $x < y\} = \{R$ und $B\}$

  $\Rightarrow \{x * x + 5 * x = 0$ und $x * x + 5 * x < y\}$

  $x := x * x + 5 * x;$

  $\{x = 0$ und $x < y\} = \{R$ und $B\}$

(Genaugenommen muss $\{R\}$ eine Verschärfung der Spezifikation des Programmes sein.)

➜ Schleife terminiert also nicht für Anfangswert $x = 0$.