

Title: Info2 (30.10.2015)

Date: Fri Oct 30 08:34:13 CET 2015

Duration: 88:46 min

Pages: 49

1.2 Korrektheit

Fragen

- Welche Programm-Eigenschaften können wir mithilfe lokal konsistenter Annotierungen garantieren ?
- Wie können wir nachweisen, dass unser Verfahren keine falschen Ergebnisse liefert ??

Zusammenfassung der Methode

- Annotiere jeden Programmpunkt mit einer Zusicherung.
- Überprüfe für jede Anweisung s zwischen zwei Zusicherungen A und B , dass A die schwächste Vorbedingung von s für B impliziert, d.h.:

$$A \Rightarrow \mathbf{WP}[[s]](B)$$

- Überprüfe entsprechend für jede Verzweigung mit Bedingung b , ob die Zusicherung A vor der Verzweigung die schwächste Vorbedingung für die Nachbedingungen B_0 und B_1 der Verzweigung impliziert, d.h.

$$A \Rightarrow \mathbf{WP}[[b]](B_0, B_1)$$

Solche Annotierungen nennen wir **lokal konsistent**.

Erinnerung (1):

- In **MiniJava** können wir ein Zustand σ aus einer **Variablen-Belegung**, d.h. einer Abbildung der Programm-Variablen auf ganze Zahlen (ihren Werten), z.B.:

$$\sigma = \{x \mapsto 5, y \mapsto -42\}$$

Erinnerung (1):

- In **MiniJava** können wir ein Zustand σ aus einer **Variablen-Belegung**, d.h. einer Abbildung der Programm-Variablen auf ganze Zahlen (ihren Werten), z.B.:

$$\sigma = \{x \mapsto 5, y \mapsto -42\}$$

- Ein Zustand σ **erfüllt** eine Zusicherung A , falls

$$A[\sigma(x)/x]_{x \in A}$$

// wir substituieren jede Variable in A durch ihren Wert in σ
eine **wahre** Aussage ist, d.h. äquivalent **true**.

Wir schreiben: $\sigma \models A$.

51

Beispiel:

$$\begin{aligned} \sigma &= \{x \mapsto 5, y \mapsto 2\} \\ A &\equiv (x > y) \\ A[5/x, 2/y] &\equiv (5 > 2) \\ &\equiv \text{true} \end{aligned}$$

52

Beispiel:

$$\begin{aligned} \sigma &= \{x \mapsto 5, y \mapsto 2\} \\ A &\equiv (x > y) \\ A[5/x, 2/y] &\equiv (5 > 2) \\ &\equiv \text{true} \end{aligned}$$

$$\begin{aligned} \sigma &= \{x \mapsto 5, y \mapsto 12\} \\ A &\equiv (x > y) \\ A[5/x, 12/y] &\equiv (5 > 12) \\ &\equiv \text{false} \end{aligned}$$

~~$\sigma \models A$~~

53

Triviale Eigenschaften:

$$\begin{aligned} \sigma \models \text{true} &\quad \text{für jedes } \sigma \\ \sigma \models \text{false} &\quad \text{für kein } \sigma \end{aligned}$$

$$\begin{aligned} \sigma \models A_1 \text{ und } \sigma \models A_2 &\quad \text{ist äquivalent zu} \\ \sigma \models A_1 \wedge A_2 & \\ \sigma \models A_1 \text{ oder } \sigma \models A_2 &\quad \text{ist äquivalent zu} \\ \sigma \models A_1 \vee A_2 & \end{aligned}$$

54

Erinnerung (2):

- Eine Programmausführung π durchläuft einen Pfad im Kontrollfluss-Graphen.
- Sie beginnt in einem Programmpunkt u_0 in einem Anfangszustand σ_0 und führt in einen Programmpunkt u_m und einen Endzustand σ_m .
- Jeder Schritt der Programm-Ausführung führt eine Aktion aus und ändert Programmpunkt und Zustand.

55

Erinnerung (2):

- Eine Programmausführung π durchläuft einen Pfad im Kontrollfluss-Graphen.
- Sie beginnt in einem Programmpunkt u_0 in einem Anfangszustand σ_0 und führt in einen Programmpunkt u_m und einen Endzustand σ_m .
- Jeder Schritt der Programm-Ausführung führt eine Aktion aus und ändert Programmpunkt und Zustand.

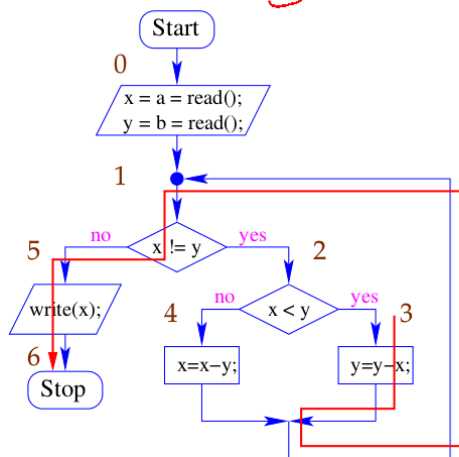
\implies Wir können π als Folge darstellen:

$$(u_0, \sigma_0) s_1 (u_1, \sigma_1) \dots s_m (u_m, \sigma_m)$$

wobei die s_i Elemente des Kontrollfluss-Graphen sind, d.h. Anweisungen oder Bedingungen ...

56

Beispiel:



57

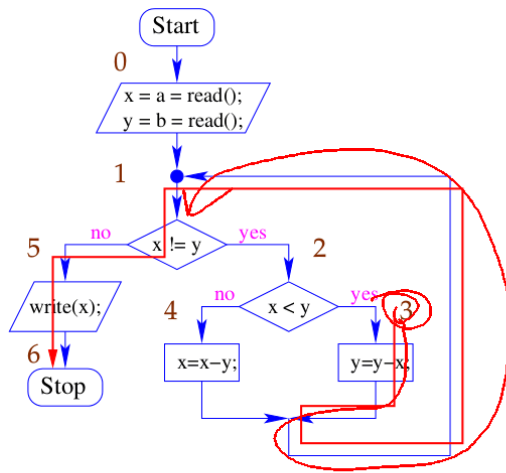
Nehmen wir an, wir starten in Punkt 3 mit $\{x \mapsto 6, y \mapsto 12\}$.

Dann ergibt sich die Programmausführung:

$$\begin{aligned} \pi = & (3, \{x \mapsto 6, y \mapsto 12\}) \quad y = y - x; \\ & (1, \{x \mapsto 6, y \mapsto 6\}) \quad !(x \neq y) \\ & (5, \{x \mapsto 6, y \mapsto 6\}) \quad \text{write}(x); \\ & (6, \{x \mapsto 6, y \mapsto 6\}) \end{aligned}$$

58

Beispiel:



57

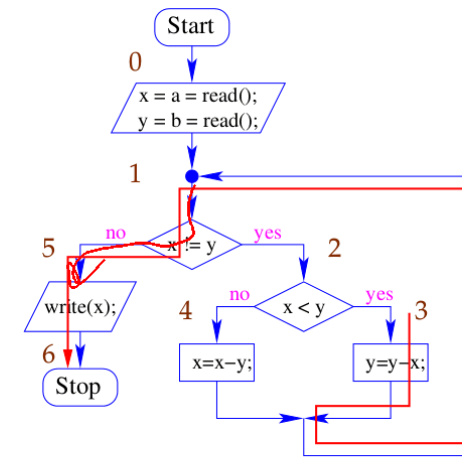
Nehmen wir an, wir starten in Punkt 3 mit $\{x \mapsto 6, y \mapsto 12\}$.

Dann ergibt sich die Programmausführung:

$\pi = (3, \{x \mapsto 6, y \mapsto 12\}) \quad y = y - x;$
 $(1, \{x \mapsto 6, y \mapsto 6\}) \quad !(x != y)$
 $(5, \{x \mapsto 6, y \mapsto 6\}) \quad \text{write}(x);$
 $(6, \{x \mapsto 6, y \mapsto 6\})$

58

Beispiel:



57

Erinnerung (2):

- Eine Programmausführung π durchläuft einen Pfad im Kontrollfluss-Graphen.
- Sie beginnt in einem Programmpunkt u_0 in einem Anfangszustand σ_0 und führt in einen Programmpunkt u_m und einen Endzustand σ_m .
- Jeder Schritt der Programm-Ausführung führt eine Aktion aus und ändert Programmpunkt und Zustand.

\implies Wir können π als Folge darstellen:

$$(u_0, \sigma_0) s_1 (u_1, \sigma_1) \dots s_m (u_m, \sigma_m)$$

wobei die s_i Elemente des Kontrollfluss-Graphen sind, d.h. Anweisungen oder Bedingungen ...

56

Nehmen wir an, wir starten in Punkt 3 mit $\{x \mapsto 6, y \mapsto 12\}$.

Dann ergibt sich die Programmausführung:

$\pi =$ (3, $\{x \mapsto 6, y \mapsto 12\}$) $y = y - x;$
(1, $\{x \mapsto 6, y \mapsto 6\}$) $!(x \neq y)$
(5, $\{x \mapsto 6, y \mapsto 6\}$) $\text{write}(x);$
(6, $\{x \mapsto 6, y \mapsto 6\}$)

58

Satz:

Sei p ein MiniJava-Programm, Sei π eine Programmausführung, die im Programmpunkt u startet und zum Programmpunkt v führt.

Annahmen:

- Die Programmpunkte von p seien lokal konsistent mit Zusicherungen annotiert. $\rightarrow G = A$
- Der Programmpunkt u sei mit A annotiert.
- Der Programmpunkt v sei mit B annotiert. $\downarrow G' = B$

Dann gilt:

Erfüllt der Anfangszustand von π die Zusicherung A , dann erfüllt der Endzustand die Zusicherung B .

60

Satz:

Sei p ein MiniJava-Programm, Sei π eine Programmausführung, die im Programmpunkt u startet und zum Programmpunkt v führt.

Annahmen:

- Die Programmpunkte von p seien lokal konsistent mit Zusicherungen annotiert.
- Der Programmpunkt u sei mit A annotiert.
- Der Programmpunkt v sei mit B annotiert.

59

Satz:

Sei p ein MiniJava-Programm, Sei π eine Programmausführung, die im Programmpunkt u startet und zum Programmpunkt v führt.

Annahmen:

- Die Programmpunkte von p seien lokal konsistent mit Zusicherungen annotiert.
- Der Programmpunkt u sei mit A annotiert.
- Der Programmpunkt v sei mit B annotiert.

Dann gilt:

Erfüllt der Anfangszustand von π die Zusicherung A , dann erfüllt der Endzustand die Zusicherung B .

60

Bemerkungen:

- Ist der Startpunkt des Programms mit **true** annotiert, dann erfüllt *jede* Programmausführung, die den Programmpunkt v erreicht, die Zusicherung an v .
- Zum Nachweis, dass eine Zusicherung A an einem Programmpunkt v gilt, benötigen wir eine lokal konsistente Annotierung mit zwei Eigenschaften:
 - (1) der Startpunkt ist mit **true** annotiert;
 - (2) Die Zusicherung an v impliziert A .



61

Bemerkungen:

- Ist der Startpunkt des Programms mit **true** annotiert, dann erfüllt *jede* Programmausführung, die den Programmpunkt v erreicht, die Zusicherung an v .
- Zum Nachweis, dass eine Zusicherung A an einem Programmpunkt v gilt, benötigen wir eine lokal konsistente Annotierung mit zwei Eigenschaften:
 - (1) der Startpunkt ist mit **true** annotiert;
 - (2) Die Zusicherung an v impliziert A .

61

Bemerkungen:

- Ist der Startpunkt des Programms mit **true** annotiert, dann erfüllt *jede* Programmausführung, die den Programmpunkt v erreicht, die Zusicherung an v .
- Zum Nachweis, dass eine Zusicherung A an einem Programmpunkt v gilt, benötigen wir eine lokal konsistente Annotierung mit zwei Eigenschaften:
 - (1) der Startpunkt ist mit **true** annotiert;
 - (2) Die Zusicherung an v impliziert A .
- Unser Verfahren gibt (vorerst) keine Garantie, dass v überhaupt erreicht wird !!!
- Falls ein Programmpunkt v mit der Zusicherung **false** annotiert werden kann, kann v **nie** erreicht werden.

62

Bemerkungen:

- Ist der Startpunkt des Programms mit **true** annotiert, dann erfüllt *jede* Programmausführung, die den Programmpunkt v erreicht, die Zusicherung an v .
- Zum Nachweis, dass eine Zusicherung A an einem Programmpunkt v gilt, benötigen wir eine lokal konsistente Annotierung mit zwei Eigenschaften:
 - (1) der Startpunkt ist mit **true** annotiert;
 - (2) Die Zusicherung an v impliziert A .
- Unser Verfahren gibt (vorerst) keine Garantie, dass v überhaupt erreicht wird !!!
- Falls ein Programmpunkt v mit der Zusicherung **false** annotiert werden kann, kann v **nie** erreicht werden.

62

Beweis:

Sei $\pi = (u_0, \sigma_0) s_1(u_1, \sigma_1) \dots s_m(u_m, \sigma_m)$

Gelte: $\sigma_0 \models A$.

Wir müssen zeigen: $\sigma_m \models B$.

Idee:

Induktion nach der Länge m der Programmausführung.

63

1.3 Optimierung




Ziel: Verringerung der benötigten Zusicherungen

Beobachtung

Hat das Programm **keine Schleifen**, können wir für jeden Programmpunkt eine hinreichende Vorbedingung **ausrechnen !!!**

65

Fazit:

- Das Verfahren nach Floyd ermöglicht uns zu beweisen, dass eine Zusicherung B bei Erreichen eines Programmpunkts stets (bzw. unter geeigneten Zusatzannahmen) gilt ...
- Zur Durchführung benötigen wir:
 - Zusicherung **true** am Startpunkt. 
 - Zusicherungen an jedem weiteren Programmpunkt. 
 - Nachweis, dass die Zusicherungen lokal konsistent sind
 \implies Logik, automatisches Beweisen 

64

Fazit:

- Das Verfahren nach Floyd ermöglicht uns zu beweisen, dass eine Zusicherung B bei Erreichen eines Programmpunkts stets (bzw. unter geeigneten Zusatzannahmen) gilt ...
- Zur Durchführung benötigen wir:
 - Zusicherung **true** am Startpunkt.
 - Zusicherungen an jedem weiteren Programmpunkt.
 - Nachweis, dass die Zusicherungen lokal konsistent sind
 \implies Logik, automatisches Beweisen

64

1.3 Optimierung

Ziel: Verringerung der benötigten Zusicherungen

Beobachtung

Hat das Programm **keine Schleifen**, können wir für jeden Programmpunkt eine hinreichende Vorbedingung **ausrechnen !!!**

65

Beispiel (Fort.)

Sei $B \equiv z = i^2 \wedge x = 2i - 1$

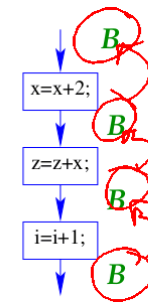
Dann rechnen wir:

$$\begin{aligned} B_1 &\equiv \mathbf{WP}[i = i+1;](B) &\equiv z = (i+1)^2 \wedge x = 2(i+1) - 1 \\ & &\equiv z = (i+1)^2 \wedge x = 2i + 1 \end{aligned}$$

67

Beispiel

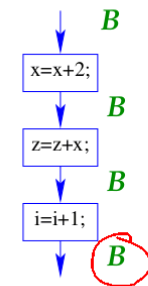
```
x = x+2;  
z = z+x;  
i = i+1;
```



66

Beispiel

```
x = x+2;  
z = z+x;  
i = i+1;
```



66

Beispiel (Fort.)

Sei $B \equiv z = i^2 \wedge x = 2i - 1$

Dann rechnen wir:

$$\begin{aligned} B_1 &\equiv \text{WP}[i = i+1;](B) &&\equiv z = (i+1)^2 \wedge x = 2(i+1) - 1 \\ &&&\equiv z = (i+1)^2 \wedge x = 2i + 1 \end{aligned}$$

67

Beispiel (Fort.)

Sei $B \equiv z = i^2 \wedge x = 2i - 1$

Dann rechnen wir:

$$\begin{aligned} B_1 &\equiv \text{WP}[i = i+1;](B) &&\equiv z = (i+1)^2 \wedge x = 2(i+1) - 1 \\ &&&\equiv z = (i+1)^2 \wedge x = 2i + 1 \\ B_2 &\equiv \text{WP}[z = z+x;](B_1) &&\equiv z + x = (i+1)^2 \wedge x = 2i + 1 \\ &&&\equiv z = i^2 \wedge x = 2i + 1 \\ B_3 &\equiv \text{WP}[x = x+2;](B_2) &&\equiv z = i^2 \wedge x + 2 = 2i + 1 \\ &&&\equiv z = i^2 \wedge x = 2i - 1 \\ &&&\equiv B \end{aligned}$$

69

Beispiel (Fort.)

Sei $B \equiv z = i^2 \wedge x = 2i - 1$

Dann rechnen wir:

$$\begin{aligned} B_1 &\equiv \text{WP}[i = i+1;](B) &&\equiv z = (i+1)^2 \wedge x = 2(i+1) - 1 \\ &&&\equiv z = (i+1)^2 \wedge x = 2i + 1 \\ B_2 &\equiv \text{WP}[z = z+x;](B_1) &&\equiv z + x = (i+1)^2 \wedge x = 2i + 1 \\ &&&\equiv z = i^2 \wedge x = 2i + 1 \end{aligned}$$

68

Idee

- Für jede Schleife wähle **einen** Programmpunkt aus.
Sinnvolle Auswahlen:
 - Vor der Bedingung;
 - Am Beginn des Rumpfs;
 - Am Ende des Rumpfs ...
- Stelle für jeden gewählten Punkt eine Zusicherung bereit
⇒ **Schleifen-Invariante**
- Für alle übrigen Programmpunkte bestimmen wir Zusicherungen mithilfe $\text{WP}[\dots]()$.

70

Beispiel

```

int a, i, x, z;
a = read();
i = 0;
x = -1;
z = 0;
while (i != a) {
    x = x+2;
    z = z+x;
    i = i+1;
}
assert(z==a*a);
write(z);

```

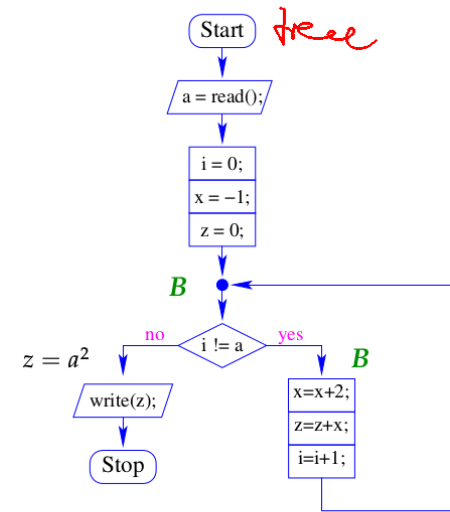
71

Wir überprüfen:

$$\begin{aligned}
 \text{WP}[i \neq a](z = a^2, B) & \\
 \equiv (i = a \wedge z = a^2) \vee (i \neq a \wedge B) & \\
 \equiv (i = a \wedge z = a^2) \vee (i \neq a \wedge z = i^2 \wedge x = 2i - 1) & \\
 \Leftarrow (i \neq a \wedge z = i^2 \wedge x = 2i - 1) \vee (i = a \wedge z = i^2 \wedge x = 2i - 1) & \\
 \equiv z = i^2 \wedge x = 2i - 1 \equiv B &
 \end{aligned}$$

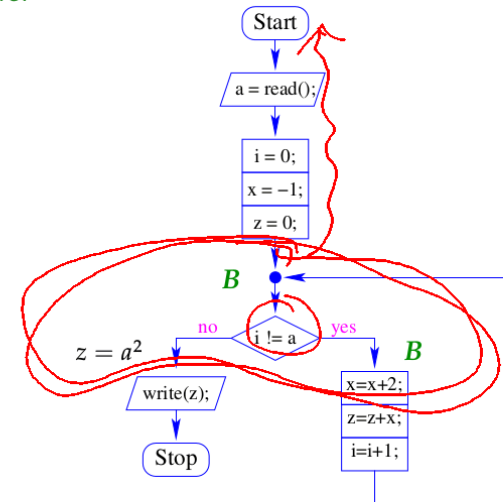
73

Beispiel



72

Beispiel



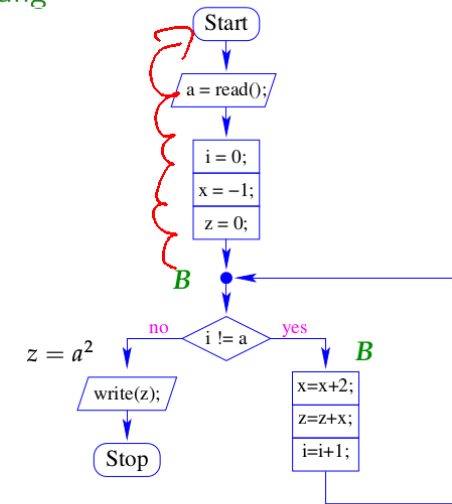
72

Wir überprüfen:

$$\begin{aligned}
 \text{WP}[i \neq a](z = a^2, B) & \\
 \equiv (i = a \wedge z = a^2) \vee (i \neq a \wedge B) & \\
 \equiv (i = a \wedge z = a^2) \vee (i \neq a \wedge z = i^2 \wedge x = 2i - 1) & \\
 \Leftarrow (i \neq a \wedge z = i^2 \wedge x = 2i - 1) \vee (i = a \wedge z = i^2 \wedge x = 2i - 1) & \\
 \equiv z = i^2 \wedge x = 2i - 1 \equiv B &
 \end{aligned}$$

73

Orientierung



74

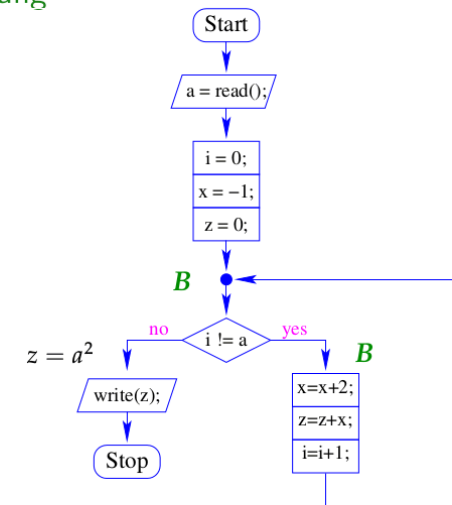
Wir überprüfen:

$$\begin{aligned}
 \text{WP}[z = 0;](B) & \equiv 0 = i^2 \wedge x = 2i - 1 \\
 & \equiv i = 0 \wedge x = -1 \\
 \text{WP}[x = -1;](i = 0 \wedge x = -1) & \equiv i = 0 \\
 \text{WP}[i = 0;](i = 0) & \equiv \text{true} \equiv 0 = 0 \\
 \text{WP}[a = \text{read}();](\text{true}) & \equiv \text{true}
 \end{aligned}$$

Handwritten note: $\forall a, \text{true} \equiv \text{true}$

75

Orientierung



74

Wir überprüfen:

$$\begin{aligned} \text{WP}[z = 0;](B) &\equiv 0 = i^2 \wedge x = 2i - 1 \\ &\equiv i = 0 \wedge x = -1 \\ \text{WP}[x = -1;](i = 0 \wedge x = -1) &\equiv i = 0 \\ \text{WP}[i = 0;](i = 0) &\equiv \text{true} \\ \text{WP}[a = \text{read}();](\text{true}) &\equiv \text{true} \end{aligned}$$

75

Beispiele

$a = b \vee$

- Das ggT-Programm terminiert nur für Eingaben a, b mit: $a > 0$ und $b > 0$.
- Das Quadrier-Programm terminiert nur für Eingaben $a \geq 0$.
- `while (true) ;` terminiert nie.
- Programme ohne Schleifen terminieren immer!

77

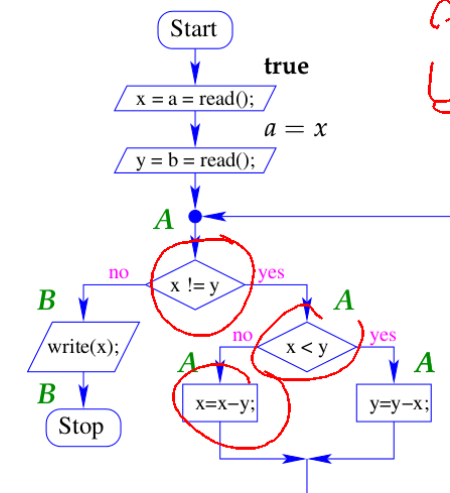
1.4 Terminierung

Problem

- Mit unserer Beweistechnik können wir nur beweisen, dass eine Eigenschaft gilt wann immer wir einen Programmpunkt erreichen !!!
- Wie können wir aber garantieren, dass das Programm immer terminiert ?
- Wie können wir eine Bedingung finden, unter der das Programm immer terminiert ??

76

Orientierung



$a = b$
 $b = 0$

46

Beispiele

- Das ggT-Programm terminiert nur für Eingaben a, b mit: $a > 0$ und $b > 0$.
- Das Quadrier-Programm terminiert nur für Eingaben $a \geq 0$.
- `while (true) ;` terminiert nie.
- Programme ohne Schleifen terminieren immer!

Beispiele

- Das ggT-Programm terminiert nur für Eingaben a, b mit: $a > 0$ und $b > 0$.
- Das Quadrier-Programm terminiert nur für Eingaben $a \geq 0$.
- `while (true) ;` terminiert nie.
- Programme ohne Schleifen terminieren immer!

Lässt sich dieses Beispiel verallgemeinern ??