

Übungen zur Vorlesung
„Grundlagen der Programm- und Systementwicklung“

Aufgabe 1 Zustandsorientierte Programme, schwächste Vorbedingungen

Betrachten Sie folgende Definitionen für die Semantik der Operatoren **skip**, **abort**, ; (Komposition), := (Zuweisung), **if ... fi** (Alternative):

$$\begin{aligned} wp(\mathbf{skip}, R) &= R \\ wp(\mathbf{abort}, R) &= \mathbf{false} \\ wp(\mathbf{P}_1; \mathbf{P}_2, R) &= wp(\mathbf{P}_1, wp(\mathbf{P}_2, R)) \\ wp(\bar{x} := \bar{e}, R) &= R[\bar{e} / \bar{x}] \\ wp(\mathbf{IF}, R) &= (\exists i \in \mathbb{N} \wedge 1 \leq i \leq n : C_i) \wedge (\forall i \in \mathbb{N} \wedge 1 \leq i \leq n : C_i \Rightarrow wp(S_i, R)) \end{aligned}$$

mit

$$\mathbf{IF} = \mathbf{if} C_1 \mathbf{then} S_1 \square C_2 \mathbf{then} S_2 \square \dots \square C_n \mathbf{then} S_n \mathbf{fi}$$

Dabei bezeichnen R und C_i Prädikate, $\mathbf{P}_1, \mathbf{P}_2$ und $S_i, 1 \leq i \leq n$, sind zustandsorientierte Programme ($n \in \mathbb{N}, 1 \leq i \leq n$), $\bar{x} = x_1, x_2, \dots, x_m$ ein Variablen-tupel und $\bar{e} = e_1, e_2, \dots, e_m$ ein gleich langes Tupel von Ausdrücken korrekten Datentyps ($m \in \mathbb{N}$). Beachten Sie, dass wir in den obigen Definitionen auf die explizite Forderung nach der Definiertheit der Ausdrücke und Prädikate verzichtet haben; wir nehmen sie vereinfachend implizit als gegeben an.

- (a) Welche Signatur hat die Funktion wp ? Geben Sie eine informelle Definition für diese Funktion. Worin unterscheidet sich wp von der ebenfalls in der Vorlesung behandelten Funktion wlp ?
- (b) Berechnen Sie folgende Prädikate:

$$\begin{aligned} &wp(x := x + 1, x \leq 1), \quad wp(\mathbf{skip}; \mathbf{P}_1, R), \quad wp(\mathbf{P}_1; \mathbf{skip}, R), \\ &wp(\mathbf{abort}; \mathbf{P}_1, R), \quad wp(\mathbf{P}_1; \mathbf{abort}, R), \\ &wp(\mathbf{if} x \geq y \mathbf{then} z := x \square y \geq x \mathbf{then} z := y \mathbf{fi}, z = \max(x, y)), \\ &wp(\mathbf{if} x \geq y \mathbf{then} z := x \square y \geq x \mathbf{then} z := y \mathbf{fi}, z = y - 1), \\ &wp(\mathbf{if} \mathbf{false} \mathbf{then} \mathbf{skip} \mathbf{fi}, \mathbf{true}) \end{aligned}$$

- (c) Was ergeben $wp(\mathbf{P}, \mathbf{false})$ und $wp(\mathbf{P}, Q \wedge R)$ für Programme \mathbf{P} und Prädikate Q und R sinnvollerweise?

- (d) Beweisen Sie die Monotonie von wp in seinem zweiten Argument:

$$(Q \Rightarrow R) \Rightarrow (wp(\mathbf{P}, Q) \Rightarrow wp(\mathbf{P}, R))$$

- (e) Beweisen Sie:

$$(wp(\mathbf{P}, Q) \vee wp(\mathbf{P}, R)) \Rightarrow wp(\mathbf{P}, Q \vee R)$$

- (f) Die Spezifikation für ein Programm \mathbf{P} über den freien Variablen x und y lautet folgendermaßen (X und Y seien Konstanten):

$$wp(\mathbf{P}, x = X \wedge y = Y) = (x = Y \wedge y = X)$$

Entwerfen Sie ein Programm \mathbf{P} , das dieser Spezifikation genügt.

Aufgabe 2 Prädikative Spezifikationen

Eine prädikative Spezifikation beschreibt die Semantik eines Programms als Relation zwischen den Zuständen σ (Anfangszustand) und σ' (Endzustand); die Relation wird dabei als Prädikat über den Variablenbelegungen in den Zuständen σ und σ' formuliert. Treten in einem Programm \mathbf{P} die freien Variablen x_1, \dots, x_k auf ($k \in \mathbb{N}$), dann bezeichnen x_1, \dots, x_k die Belegungen dieser Variablen *vor* der Ausführung von \mathbf{P} ; entsprechend bezeichnen x'_1, \dots, x'_k die Belegungen der Variablen *nach* der Ausführung von \mathbf{P} .

- (a) Formulieren Sie die Spezifikation aus Aufgabe 1(f) als prädikative Spezifikation.
- (b) Geben Sie eine prädikative Spezifikation für ein Programm \mathbf{P} über den Variablen x und y an, sodass die Variablen nach der Ausführung von \mathbf{P} so permutiert sind, dass $x \leq y$ gilt. Entwerfen Sie ein Programm, das dieser Spezifikation genügt.
- (c) Formulieren Sie eine prädikative Spezifikation für ein Programm, das ein gegebenes Array permutiert.
- (d) Wie hängen die Methode der schwächsten Vorbedingungen und prädikative Spezifikationen formal zusammen?

Aufgabe 3 Zusicherungsmethode

Mittels der Zusicherungsmethode lassen sich ebenfalls Programmspezifikationen angeben. Hier werden zu einem Programm die Vorbedingung (precondition), die vor der Ausführung des Programms erfüllt sein muß, sowie die Nachbedingung (postcondition), die nach der Ausführung des Programms gültig ist angegeben. Wir schreiben etwa:

$$\{Pre\} \mathbf{S} \{Post\}$$

mit der Bedeutung

„Wird die Ausführung von **S** in einem Zustand gestartet, in dem die Zusicherung *Pre* gilt, dann terminiert **S** und danach gilt die Zusicherung *Post*.“

- (a) Wie hängen die Zusicherungsmethode und die Methode der schwächsten Vorbedingungen formal zusammen?
 (b) Annotieren Sie folgendes Programm durch Zusicherungen:

$$t := x; x := y; y := t$$

- (c) Zum Nachweis von Eigenschaften zustandsorientierter Programme lässt sich der *Zusicherungskalkül* verwenden. Seine grundlegenden Axiome lauten wie folgt:

| | |
|---|------------------------|
| $\{R\} \mathbf{skip} \{R\}$ | <i>leere Anweisung</i> |
| $\{R[E/x]\} x := E \{R\}$ | <i>Zuweisung</i> |
| $\frac{Q1 \Rightarrow Q \quad \{Q\} S \{R\} \quad R \Rightarrow R1}{\{Q1\} S \{R1\}}$ | <i>Abschwächung</i> |
| $\frac{\{Q\} S1 \{R1\} \quad \{R1\} S2 \{R\}}{\{Q\} S1 ; S2 \{R\}}$ | <i>Komposition</i> |
| $\frac{\{C \wedge Q\} S1 \{R\} \quad \{\neg C \wedge Q\} S2 \{R\}}{\{Q\} \mathbf{if} C \mathbf{then} S1 \square \neg C \mathbf{then} S2 \mathbf{fi} \{R\}}$ | <i>Alternative</i> |
| $\frac{\{C \wedge R\} S \{R\}}{\{R\} \mathbf{while} C \mathbf{do} S \mathbf{od} \{R \wedge \neg C\}}$ | <i>Wiederholung</i> |

Begründen Sie die Plausibilität dieser Axiome informell, jedoch auf Basis der entsprechenden Regeln für die Methode der schwächsten Vorbedingungen.

Aufgabe 4 Programmverifikation mittels Zusicherungsmethode

Im folgenden setzen wir die Zusicherungsmethode ein, um die Korrektheit eines zustandsorientierten Programms zu beweisen. Angegeben sind dabei das Programm selbst, sowie Vor- und Nachbedingung:

```

{xs = s ∧ isempty(ys)}
while not isempty(xs) do
  if first(xs) = x then skip
  □  $\neg(\text{first}(xs) = x)$  then ys := first(xs) ◦ ys
  fi
  xs := rest(xs)
od
{freq(x, ys) = 0 ∧ (∀z : z ≠ x ⇒ (freq(z, ys) = freq(z, s)))}
    
```

Dabei nehmen wir an, dass *xs* und *ys* vom Typ **var Seq** α und *x* sowie *z* vom Typ α sind. *freq* sei eine Funktion, deren Resultat angibt, wie häufig das als erster Parameter übergebene Element im zweiten Parameter (einer Sequenz) vorkommt. Im folgenden Bezeichnen wir die hier gegebene Vorbedingung mit *P*, das Programm mit **S** und die Nachbedingung mit *Q*.

- (a) Was besagt die Behauptung $\{P\} \mathbf{S} \{Q\}$? Ist sie plausibel?
 (b) Zerteilen Sie das Programmstück in syntaktische Einheiten, zu denen es Regeln im Zusicherungskalkül gibt und bezeichnen Sie diese Einheiten.
 (c) Was lässt sich bei einem Schleifendurchgang über den Ausdruck $\text{freq}(z, xs) + \text{freq}(z, ys)$ sagen?
 (d) Geben Sie eine Schleifeninvariante *R* an. Lässt sich mit Hilfe von *R* die Behauptung beweisen?
 (e) Beweisen Sie die Invarianz von *R*.