Theoretische Informatik
Übungsblatt 5 (2025W)
Lösungsvorschlag

In den ersten fünf Aufgaben steht Σ für das folgende Programm.

$y := 2 * x + 1$;
if $y > 0$ **then**
  abort
else
  $x := 2 * y$
fi

Aufgabe 5.1

Zeigen Sie, dass das Programm Σ nach Hinzufügen einiger Klammernpaare ein syntaktisch korrektes $\mathsf{SIMPLE}(\mathbb{Z})$-Programm ist.

Anleitung: Geben Sie eine Parallelableitung an.

Lösung 5.1

Bei einer Parallelableitung werden ausgehend von der Startvariablen in jedem Ableitungsschritt alle Nonterminale mittels einer geeigneten Produktion ersetzt.

$$
\begin{aligned}
&\mathit{Programm}\\
&\Rightarrow_P \mathit{Programm}\ \text{„}\mathbf{;}\text{“}\ \mathit{Programm}\\
&\Rightarrow_P \mathit{Var}\ \text{„}:=\text{“}\ \mathit{Term}\ \text{„}\mathbf{;\ if}\text{“}\ \mathit{Term}\ \text{„}\mathbf{then}\text{“}\ \mathit{Programm}\ \text{„}\mathbf{else}\text{“}\ \mathit{Programm}\ \text{„}\mathbf{fi}\text{“}\\
&\Rightarrow_P \text{„}y := (\text{“}\ \mathit{Term}\ \mathit{BinOp}\ \mathit{Term}\ \text{„})\mathbf{;\ if}\ (\text{“}\ \mathit{Term}\ \mathit{BinOp}\ \mathit{Term}\ \text{„})\ \mathbf{then\ abort\ else}\text{“}\ \mathit{Var}\ \text{„}:=\text{“}\ \mathit{Term}\ \text{„}\mathbf{fi}\text{“}\\
&\Rightarrow_P \text{„}y := ((\text{“}\ \mathit{Term}\ \mathit{BinOp}\ \mathit{Term}\ \text{„}) + \text{“}\ \mathit{Const}\ \text{„})\mathbf{;}\ \text{“}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{„}\mathbf{if}\ (\text{“}\ \mathit{Var}\ \text{„}>\text{“}\ \mathit{Const}\ \text{„})\ \mathbf{then\ abort\ else}\ x := (\text{“}\ \mathit{Term}\ \mathit{BinOp}\ \mathit{Term}\ \text{„})\ \mathbf{fi}\text{“}\\
&\Rightarrow_P \text{„}y := ((\text{“}\ \mathit{Const}\ \text{„}*\text{“}\ \mathit{Var}\ \text{„}) + 1)\mathbf{;\ if}\ (y > 0)\ \mathbf{then\ abort\ else}\ x := (\text{“}\ \mathit{Const}\ \text{„}*\text{“}\ \mathit{Var}\ \text{„})\ \mathbf{fi}\text{“}\\
&\Rightarrow_P \text{„}y := ((2 * x) + 1)\mathbf{;\ if}\ (y > 0)\ \mathbf{then\ abort\ else}\ x := (2 * y)\ \mathbf{fi}\text{“}
\end{aligned}
$$

Aufgabe 5.2

(a) Sei σ ein Zustand mit $\sigma(x) = 0$. Berechnen Sie $[\Sigma]\sigma$ mittels der strukturellen operationalen Semantik von $\mathsf{SIMPLE}(\mathbb{Z})$.

(b) Sei σ ein Zustand mit $\sigma(x) = -1$. Berechnen Sie $[\Sigma]\sigma$ mittels der natürlichen Semantik von $\mathsf{SIMPLE}(\mathbb{Z})$.

Lösung 5.2

(a) Der Zustand σ' ist das Ergebnis von $[\Sigma]\sigma$, wenn er sich mittels $(\Sigma, \sigma) \overset{*}{\Rightarrow} \sigma'$ berechnen lässt.

$(\Sigma, \sigma) = (y \coloneqq 2{*}x{+}1$**; if** $y{>}0$ **then abort else** $x \coloneqq 2{*}y$ **fi**$, \sigma)$

$(y \coloneqq 2{*}x{+}1, \sigma) \Rightarrow \sigma_1$, wobei $\sigma_1(y) = [2{*}x{+}1]\sigma = 2\sigma(x) + 1 = 1$

$\sigma_1(v) = \sigma(v) \quad$ für $v \neq y$

$\Rightarrow$ (**if** $y{>}0$ **then abort else** $x \coloneqq 2{*}y$ **fi**$, \sigma_1)$

$[y > 0]\sigma_1 = (\sigma_1(y) > 0) = (1 > 0) = 1$

$\Rightarrow$ (**abort**$, \sigma_1)$

Die Berechnung bleibt an dieser Stelle stecken, da keine Auswertungsregeln für **abort** definiert sind. Da es kein σ' gibt, sodass $(\Sigma, \sigma) \overset{*}{\Rightarrow} \sigma'$ gilt, ist $[\Sigma]\sigma$ undefiniert.

(b) $[\Sigma]\,\sigma = [y \coloneqq 2{*}x{+}1$**; if** $y{>}0$ **then abort else** $x \coloneqq 2{*}y$ **fi**$]\,\sigma$

$= [$**if** $y{>}0$ **then abort else** $x \coloneqq 2{*}y$ **fi**$]\,[y \coloneqq 2{*}x{+}1]\,\sigma$

$= [$**if** $y{>}0$ **then abort else** $x \coloneqq 2{*}y$ **fi**$]\,\sigma_1$, wobei $\sigma_1(y) = [2{*}x{+}1]\sigma = 2\sigma(x) + 1 = -1$

$\sigma_1(v) = \sigma(v) \quad$ für $v \neq y$

$[y > 0]\sigma_1 = (\sigma_1(y) > 0) = (-1 > 0) = 0$

$= [x \coloneqq 2{*}y]\,\sigma_1$

$= \sigma_2$, wobei $\sigma_2(x) = [2{*}y]\sigma_1 = 2\sigma_1(y) = -2$

$\sigma_2(v) = \sigma_1(v) \quad$ für $v \neq x$

Aufgabe 5.3

Wir betrachten die Korrektheitsaussage $\{x > y\}\,\Sigma\,\{x \leq y\}$.

(a) Zeigen Sie, dass die Aussage partiell korrekt ist, indem Sie die Korrektheitsaussage mit den Regeln und Axiomen des Hoare-Kalküls ableiten. Argumentieren Sie, warum die auftretenden Formeln gültig sind.

Anleitung: Die Interpolante für die Anwendung der Regel (ha) lässt sich bestimmen, indem man für die erste Anweisung das Axiom (zw′) verwendet.

(b) Zeigen Sie, dass die Korrektheitsaussage nicht total korrekt ist.

Anleitung: Geben Sie ein Gegenbeispiel an und argumentieren Sie, warum daraus folgt, dass die Korrektheitsaussage nicht wahr ist.

Lösung 5.3

(a) Die folgende Ableitung lässt sich von oben nach unten als eine Reihe von Regelanwendungen lesen, bei denen von wahren Korrektheitsaussagen und gültigen Formeln auf komplexere wahre Korrektheitsaussagen geschlossen wird. Konstruiert wird die Ableitung aber von unten nach oben, beginnend mit der zu beweisenden Korrektheitsaussage.

$$\dfrac{\dfrac{\{x > y\}\, y \coloneqq 2{*}x{+}1\,\{G\}\ {}_{(\mathrm{zw}')} \qquad \dfrac{\{G\wedge y{>}0\}\,\textbf{abort}\,\{x \leq y\}\ {}_{(\mathrm{ab})} \qquad \dfrac{(G \wedge y \leq 0) \Rightarrow (x \leq y)\left[{2*y}\atop{x}\right] \quad \{(x \leq y)\left[{2*y}\atop{x}\right]\}\, x \coloneqq 2{*}y\,\{x \leq y\}\ {}_{(\mathrm{zw})}}{\{G\wedge y \leq 0\}\, x \coloneqq 2{*}y\,\{x \leq y\}}\ {}_{(\mathrm{imp})}}{\{G\}\,\textbf{if}\ y{>}0\ \textbf{then abort else}\ x \coloneqq 2{*}y\ \textbf{fi}\,\{x \leq y\}}\ {}_{(\mathrm{if})}}{\{x > y\}\, y \coloneqq 2{*}x{+}1;\ \textbf{if}\ y{>}0\ \textbf{then abort else}\ x \coloneqq 2{*}y\ \textbf{fi}\,\{x \leq y\}}\ {}_{(\mathrm{ha})}$$

Wir wählen G so, dass $\{x > y\}\, y \coloneqq 2{*}x{+}1\,\{G\}$ eine Instanz des Axioms zw′ bildet, d.h., wir wählen

$$G = \exists y'(x > y' \wedge y = 2{*}x{+}1) = (y = 2{*}x{+}1) \wedge \underbrace{\exists y'\, x > y'}_{\text{wahr}} = (y = 2{*}x{+}1)\ .$$

Somit ist die Gültigkeit der folgenden Implikation zu zeigen.

$$\begin{aligned}(G \wedge y \leq 0) &\Rightarrow (x \leq y)\left[{2*y}\atop{x}\right]\\ (y = 2{*}x{+}1 \wedge y \leq 0) &\Rightarrow 2{*}y \leq y\end{aligned}$$

Die rechte Seite der Implikation ist äquivalent zu $y \leq 0$ (Subtraktion von y).

$$(y = 2{*}x{+}1 \wedge y \leq 0) \Rightarrow y \leq 0$$

Diese Formel ist eine Tautologie der Form $(\cdots \wedge F) \Rightarrow F$, also gültig.

Insgesamt haben wir die Korrektheitsaussage in der letzten Zeile der Ableitung mit den Regeln des Hoare-Kalküls für partielle Korrektheit aus den Axiomen (zw), (zw′), (ab) und einer gültigen Formel hergeleitet. Aus der Korrektheit des Hoare-Kalküls folgt, dass diese Aussage daher partiell korrekt ist.

(b) Ein Gegenbeispiel für die totale Korrektheit einer Korrektheitsaussage besteht aus einer zulässigen Eingabe, für die das Programm entweder kein Ergebnis liefert oder das Ergebnis nicht die Nachbedingung erfüllt. Da die Aussage partiell korrekt ist, erfüllt jedes Ergebnis einer zulässigen Eingabe die Nachbedingung. Wir benötigen daher eine zulässige Eingabe σ, für die das Ergebnis des Programms nicht definiert ist. Wir wählen $\sigma(x) = 0$ and $\sigma(y) = -1$ und erhalten:

- σ ist eine zulässige Eingabe, da sie die Vorbedingung erfüllt.

$$[x > y]\,\sigma = \sigma(x) > \sigma(y) = 0 > -1 = 1$$

- Das Ergebnis $[\Sigma]\,\sigma$ ist nicht definiert.

$$[\Sigma]\,\sigma = \cdots = [\textbf{abort}]\,\sigma_1 = \text{undefiniert, siehe Aufgabe 2(a)}$$

Die Korrektheitsaussage ist daher nicht wahr bzgl. totaler Korrektheit, ist also nicht „total korrekt“.

Aufgabe 5.4

Zeigen Sie, dass die Aussage $\{x > y\}\,\Sigma\,\{x \leq y\}$ partiell korrekt ist, indem Sie wahlweise die schwächste liberale Vorbedingung (wlp) oder die stärkste Nachbedingung (sp) des Programms berechnen. Argumentieren Sie, warum die auftretende Implikation gültig ist.

Lösung 5.4

Schwächste liberale Vorbedingungen: Wir berechnen $\mathsf{wlp}(\Sigma, x \leq y)$ und zeigen, dass diese Formel von der Vorbedingung impliziert wird.

$$\begin{aligned}
\mathsf{wlp}(\Sigma, x \leq y) &= \mathsf{wlp}(y \coloneqq 2{*}x{+}1\textbf{; if } y{>}0 \textbf{ then abort else } x \coloneqq 2{*}y \textbf{ fi}, x \leq y)\\
&= \mathsf{wlp}(y \coloneqq 2{*}x{+}1,\ \mathsf{wlp}(\textbf{if } y{>}0 \textbf{ then abort else } x \coloneqq 2{*}y \textbf{ fi}, x \leq y))\\
&= \mathsf{wlp}(y \coloneqq 2{*}x{+}1,\ (y{>}0 \Rightarrow \mathsf{wlp}(\textbf{abort}, x{\leq}y)) \wedge (y{\leq}0 \Rightarrow \mathsf{wlp}(x \coloneqq 2{*}y, x \leq y)))\\
&= \mathsf{wlp}(y \coloneqq 2{*}x{+}1,\ (y{>}0 \Rightarrow 1) \wedge (y{\leq}0 \Rightarrow 2y{\leq}y))\\
&= \mathsf{wlp}(y \coloneqq 2{*}x{+}1,\ (y{>}0 \Rightarrow 1) \wedge (y{\leq}0 \Rightarrow y{\leq}0))\\
&= \mathsf{wlp}(y \coloneqq 2{*}x{+}1,\ 1 \wedge 1)\\
&= \mathsf{wlp}(y \coloneqq 2{*}x{+}1,\ 1)\\
&= 1
\end{aligned}$$

$$\begin{aligned}
x > y &\Rightarrow \mathsf{wlp}(\Sigma, x \leq y)\\
x > y &\Rightarrow 1
\end{aligned}$$

Die Implikation in der letzten Zeile ist offenbar gültig.

Stärkste Nachbedingung: Wir berechnen $\mathsf{sp}(x > y, \Sigma)$ und zeigen, dass diese Formel die Nachbedingung impliziert.

$$\begin{aligned}
\mathsf{sp}(x > y, \Sigma) &= \mathsf{sp}(x > y,\ y \coloneqq 2{*}x{+}1\textbf{; if } y{>}0 \textbf{ then abort else } x \coloneqq 2{*}y \textbf{ fi})\\
&= \mathsf{sp}(\mathsf{sp}(x > y,\ y \coloneqq 2{*}x{+}1),\ \textbf{if } y{>}0 \textbf{ then abort else } x \coloneqq 2{*}y \textbf{ fi})\\
&= \mathsf{sp}(\exists y'(x > y' \wedge y{=}2{*}x{+}1),\ \textbf{if } y{>}0 \textbf{ then abort else } x \coloneqq 2{*}y \textbf{ fi})\\
&= \mathsf{sp}(y{=}2{*}x{+}1,\ \textbf{if } y{>}0 \textbf{ then abort else } x \coloneqq 2{*}y \textbf{ fi})\\
&= \mathsf{sp}(y{=}2{*}x{+}1 \wedge y{>}0,\ \textbf{abort}) \vee \mathsf{sp}(y{=}2{*}x{+}1 \wedge y{\leq}0,\ x \coloneqq 2{*}y)\\
&= 0 \vee \mathsf{sp}(y{=}2{*}x{+}1 \wedge y{\leq}0,\ x \coloneqq 2{*}y)\\
&= \mathsf{sp}(y{=}2{*}x{+}1 \wedge y{\leq}0,\ x \coloneqq 2{*}y)\\
&= y{\leq}0 \wedge x = 2y \wedge \exists x'\, y{=}2x'{+}1\\
&= y{\leq}0 \wedge x = 2y \wedge y \text{ ungerade}
\end{aligned}$$

$$\begin{aligned}
\mathsf{sp}(x > y, \Sigma) &\Rightarrow x{\leq}y\\
y{\leq}0 \wedge x = 2y \wedge y \text{ ungerade} &\Rightarrow x{\leq}y
\end{aligned}$$

Aus $y{\leq}0$ folgt $2y{\leq}y$ (Addition von y auf beiden Seiten). Mit der zweiten Prämisse erhalten wir $x \leq y$, die Implikation ist also gültig.

Aufgabe 5.5
Zeigen Sie, dass die Aussage $\{x < -4\}\,\Sigma\,\{y > x+5\}$ total korrekt ist. Führen Sie den Korrektheitsbeweis, indem Sie das Programm mithilfe der Annotierungsregeln um Zustandsbeschreibungen ergänzen und die Gültigkeit der auftretenden Implikationen argumentieren.

$\{\, x < -4 \,\}$
$y := 2 * x + 1$;
if $y > 0$ **then**
  abort
else
  $x := 2 * y$
fi
$\{\, y > x + 5 \,\}$

Lösung 5.5
Wir beginnen mit der zu beweisenden Korrektheitsaussage und fügen schrittweise Zustandsbeschreibungen hinzu. Wir nummerieren die Formeln in der Reihenfolge des Hinzufügens.

$\{\, \mathit{Pre}\colon x < -4 \,\}$
zw↑ $\{\, F_7\colon \big((y > 0 \wedge 0) \vee (y \leq 0 \wedge y > 2y + 5)\big)\big[{}^{2x+1}_{y}\big] \,\}$
$y := 2 * x + 1$;
if↑ $\{\, F_6\colon (y > 0 \wedge 0) \vee (y \leq 0 \wedge y > 2y + 5) \,\}$
if $y > 0$ **then**
ab $\{\, F_1\colon 0 \,\}$
  abort
ab $\{\, F_2\colon 0 \,\}$
fi↑ $\{\, F_3\colon y > x + 5 \,\}$
else
zw↑ $\{\, F_5\colon y > 2y + 5 \,\}$
  $x := 2 * y$
fi↑ $\{\, F_4\colon y > x + 5 \,\}$
fi
$\{\, \mathit{Post}\colon y > x + 5 \,\}$

Wir müssen die Gültigkeit von zwei Implikationen zeigen.

- $\mathit{Pre} \Rightarrow F_7$:
  $x < -4 \Rightarrow \big((y > 0 \wedge 0) \vee (y \leq 0 \wedge y > 2y + 5)\big)\big[{}^{2x+1}_{y}\big]$
  $x < -4 \Rightarrow \big(0 \vee (y \leq 0 \wedge y < -5)\big)\big[{}^{2x+1}_{y}\big]$
  $x < -4 \Rightarrow (y < -5)\big[{}^{2x+1}_{y}\big]$
  $x < -4 \Rightarrow 2x + 1 < -5$
  $x < -4 \Rightarrow x < -3$
  Diese Implikation ist offenbar gültig über $\mathbb{Z}$.
- $F_2 \Rightarrow F_3$, d.h., $0 \Rightarrow y > x + 5$. Implikationen der Form $0 \Rightarrow G$ sind für beliebiges G gültig.

Damit haben wir die totale Korrektheit mit Hilfe des Annotierungskalküls gezeigt.

Aufgabe 5.6

Zeigen Sie, dass die folgende Aussage total korrekt ist. Welche Funktion berechnet das Programm, wenn x_0 die Ein- und y die Ausgabe ist?

Anleitung: Verwenden Sie die Formel $x2^y \leq x_0 < (x+1)2^y \wedge x \geq 1$ als Invariante und den Term x als Variante. Beachten Sie, dass / die ganzzahlige Division bezeichnet und damit das Vereinfachen von Brüchen, etwa von $2\frac{x}{2}$ zu x, nicht ohne Weiteres möglch ist. Benützen Sie stattdessen die Beziehung $2\frac{x}{2} \leq x \leq 2\frac{x}{2} + 1$.

$\{\, x = x_0 \wedge x_0 \geq 1 \,\}$
$y \coloneqq 0$;
while $x > 1$ **do**
  $y \coloneqq y + 1$;
  $x \coloneqq x/2$
od;
$\{\, 2^y \leq x_0 < 2^{y+1} \,\}$

Lösung 5.6

Wir annotieren die Korrektheitsaussage schrittweise mit Zustandsbeschreibungen innerhalb des Programms. Wir nummerieren die Formeln in der Reihenfolge des Hinzufügens.

$\{\, \mathit{Pre}\colon x = x_0 \wedge x_0 \geq 1 \,\}$
zw↑ $\{\, F_7\colon \mathit{Inv}\binom{0}{y} \,\}$
$y \coloneqq 0$;
wht $\{\, F_1\colon \mathit{Inv} \,\}$
while $x > 1$ **do**
wht $\{\, F_2\colon \mathit{Inv} \wedge x > 1 \wedge t = t_0 \,\}$
zw↑ $\{\, F_6\colon (\mathit{Inv} \wedge 0 \leq t < t_0)\binom{x/2}{x}\binom{y+1}{y} \,\}$
  $y \coloneqq y + 1$;
zw↑ $\{\, F_5\colon (\mathit{Inv} \wedge 0 \leq t < t_0)\binom{x/2}{x} \,\}$
  $x \coloneqq x/2$
wht $\{\, F_3\colon \mathit{Inv} \wedge 0 \leq t < t_0 \,\}$
od;
wht $\{\, F_4\colon \mathit{Inv} \wedge \neg(x > 1) \,\}$
$\{\, \mathit{Post}\colon 2^y \leq x_0 < 2^{y+1} \,\}$

Nun müssen wir noch die Gültigkeit der Formeln $\mathit{Pre} \Rightarrow F_7$, $F_2 \Rightarrow F_6$ und $F_4 \Rightarrow \mathit{Post}$ argumentieren, wobei wir Invariante und Variante gemäß dem Hinweis in der Angabe wählen.

- $\mathit{Pre} \Rightarrow F_7$:

  $x = x_0 \wedge x_0 \geq 1 \Rightarrow \mathit{Inv}\binom{0}{y}$

  $x = x_0 \wedge x_0 \geq 1 \Rightarrow x2^0 \leq x_0 < (x+1)2^0 \wedge x \geq 1$

  $x = x_0 \wedge x_0 \geq 1 \Rightarrow x \leq x_0 < x + 1 \wedge x \geq 1$

  Wir müssen zeigen, dass die Bedingungen auf der rechten Seite erfüllt sind, wobei wir voraussetzen können, dass die linke Seite der Implikation gilt.

$x \leq x_0$	gilt wegen der Prämisse $x = x_0$
$x_0 < x + 1$	folgt aus $x_0 < x_0 + 1$ und der Prämisse $x = x_0$
$x \geq 1$	folgt aus den Prämissen $x = x_0$ und $x_0 \geq 1$

- $F_2 \Rightarrow F_6$:

  $\mathit{Inv} \wedge x > 1 \wedge t = t_0 \Rightarrow (\mathit{Inv} \wedge 0 \leq t < t_0)\binom{x/2}{x}\binom{y+1}{y}$

  $\cdots \Rightarrow (x2^y \leq x_0 < (x+1)2^y \wedge x \geq 1 \wedge 0 \leq x < t_0)\binom{x/2}{x}\binom{y+1}{y}$

  $x2^y \leq x_0 < (x+1)2^y \wedge x \geq 1 \wedge x > 1 \wedge x = t_0$

  $\Rightarrow \frac{x}{2}2^{y+1} \leq x_0 < (\frac{x}{2}+1)2^{y+1} \wedge \frac{x}{2} \geq 1 \wedge 0 \leq \frac{x}{2} < t_0$

  Diese Implikation ist gültig, weil:

$\frac{x}{2}2^{y+1} \leq x_0$	Es gilt $2\frac{x}{2} \leq x$ (siehe Angabe). Multiplikation mit 2^y liefert $\frac{x}{2}2^{y+1} \leq x2^y$. Zusammen mit der Prämisse $x2^y \leq x_0$ erhalten wir die gesuchte Aussage.
$x_0 < (\frac{x}{2}+1)2^{y+1}$	Es gilt $x \leq 2\frac{x}{2}+1$ (siehe Angabe). Addition von 1 und Multiplikation mit 2^y auf beiden Seiten liefert $(x+1)2^y \leq (2\frac{x}{2}+2)2^y$. Die rechte Seite vereinfacht zu $2(\frac{x}{2}+1)2^y = (\frac{x}{2}+1)2^{y+1}$. Kombination der Ungleichung $(x+1)2^y \leq (\frac{x}{2}+1)2^{y+1}$ mit der Prämisse $x_0 < (x+1)2^y$ liefert die gesuchte Aussage $x_0 < (\frac{x}{2}+1)2^{y+1}$.
$\frac{x}{2} \geq 1$	Folgt aus der Prämisse $x > 1$.
$0 \leq \frac{x}{2}$	Folgt aus der Prämisse $x > 1$.
$\frac{x}{2} < t_0$	Äquivalent zu $\frac{x}{2} < x$ (Prämisse $x = t_0$), gilt wegen Prämisse $x > 1$.

- $F_4 \Rightarrow \mathit{Post}$:

$$\mathit{Inv} \wedge \neg(x > 1) \Rightarrow 2^y \leq x_0 < 2^{y+1}$$
$$x2^y \leq x_0 < (x+1)2^y \wedge x \geq 1 \wedge \neg(x > 1) \Rightarrow 2^y \leq x_0 < 2^{y+1}$$

  Aus den Prämissen $x \geq 1$ und $\neg(x > 1)$ folgt $x = 1$. Damit erhalten wir aus der ersten Prämisse die Konklusion $2^y \leq x_0 < 2^{y+1}$.

Damit haben wir gezeigt, dass die Aussage wahr bzgl. totaler Korrektheit ist.
Um die berechnete Funktion zu bestimmen, formen wir die Nachbedingung um und drücken y als Funktion von x_0 aus.

$2^y \leq x_0 < 2^{y+1}$	
$\log_2(2^y) \leq \log_2(x_0) < \log_2(2^{y+1})$	da log streng monoton ist
$y \leq \log_2(x_0) < y+1$	
$y = \lfloor \log_2(x_0) \rfloor$	für $y \in \mathbb{Z}$ gilt: $y = \lfloor x \rfloor \Leftrightarrow y \leq x \leq y+1$

Das Programm berechnet also den ganzzahligen Teil des Zweierlogarithmus von x_0.

Aufgabe 5.7
In der Vorlesung wurde die Weakest Liberal Precondition einer Schleife folgendermaßen angegeben:

$$\mathsf{wlp}(\textbf{while } e \textbf{ do } \Pi \textbf{ od}, G) = \forall i\,(i \geq 0 \Rightarrow F_i)$$
$$\text{wobei } F_0 = e \vee G \text{ und } F_{i+1} = \neg e \vee \mathsf{wlp}(\Pi, F_i)$$

Argumentieren Sie, warum sich die Weakest Liberal Precondition einer Schleife so berechnen lässt.

Anleitung: Sie können die folgenden Zusammenhänge voraussetzen und verwenden.

(1) $\mathsf{wp}(\textbf{while } e \textbf{ do } \Pi \textbf{ od}, H) = \exists i\,(i \geq 0 \wedge E_i)$, wobei $E_0 = \neg e \wedge H$ und $E_{i+1} = e \wedge \mathsf{wp}(\Pi, E_i)$

(2) $\mathsf{wlp}(\Pi, G) = \neg\mathsf{wp}(\Pi, \neg G)$

(3) $\mathsf{wp}(\Pi, G) = \neg\mathsf{wlp}(\Pi, \neg G)$

(1) wurde in der Vorlesung erklärt; hier sind lediglich G und F_i umbenannt in H und E_i, um Verwechslungen mit G und F_i in der Definition von wlp zu vermeiden.
Die Gleichungen (2) und (3) ergeben sich, wenn man die Menge aller Zustände in folgende disjunkte Gruppen teilt.

(A) Die Zustände, für die das Programm Π nicht terminiert.

(B) Die Zustände, für die das Programm Π terminiert und sein Ergebnis die Formel G erfüllt.

(C) Die Zustände, für die das Programm Π terminiert und sein Ergebnis die Formel $\neg G$ erfüllt.

$\mathsf{wlp}(\Pi, G)$ repräsentiert die Vereinigung von (A) und (B), während $\mathsf{wp}(\Pi, \neg G)$ der Menge (C) entspricht. Da die ersten beiden Mengen das Komplement der dritten sind, ist $\mathsf{wlp}(\Pi, G)$ die Negation von $\mathsf{wp}(\Pi, \neg G)$. Analog ergibt sich Gleichung (3), wenn man (A) und (C) durch $\mathsf{wlp}(\Pi, \neg G)$ und (B) durch $\mathsf{wp}(\Pi, G)$ beschreibt.

Lösung 5.7
Wir führen wlp auf wp zurück und formen die Ausdrücke so um, dass sie der zu beweisenden Beziehung entsprechen.

$$\begin{aligned}
\mathsf{wlp}(\textbf{while } e \textbf{ do } \Pi \textbf{ od}, G) &= \neg\mathsf{wp}(\textbf{while } e \textbf{ do } \Pi \textbf{ od}, \neg G) && \text{Gleichung (2) mit } H = \neg G \\
&= \neg(\exists i\,(i \geq 0 \wedge E_i)) && \text{Gleichung (1)} \\
&= \forall i\,\neg(i \geq 0 \wedge E_i) && \text{wegen } \neg\exists x A = \forall x \neg A \\
&= \forall i\,(\neg(i \geq 0) \vee \neg E_i) && \text{de Morgan} \\
&= \forall i\,(i \geq 0 \Rightarrow \neg E_i) && \text{wegen } A \Rightarrow B = \neg A \vee B
\end{aligned}$$

wobei

$$\begin{aligned}
E_0 &= \neg e \wedge \neg G && \text{Gleichung (2) mit } H = \neg G \\
&= \neg(e \vee G) && \text{de Morgan} \\
\neg E_0 &= e \vee G && \text{Negieren beider Seiten}
\end{aligned}$$

und

$$\begin{aligned}
E_{i+1} &= e \wedge \mathsf{wp}(\Pi, E_i) && \text{Gleichung (2)} \\
&= e \wedge \neg\mathsf{wlp}(\Pi, \neg E_i) && \text{Gleichung (3)} \\
&= \neg(\neg e \vee \mathsf{wlp}(\Pi, \neg E_i)) && \text{de Morgan} \\
\neg E_{i+1} &= \neg e \vee \mathsf{wlp}(\Pi, \neg E_i) && \text{Negieren beider Seiten}
\end{aligned}$$

Wenn wir nun überall $\neg E_i$ in F_i umbenennen, erhalten wir die gesuchte Beziehung.

Aufgabe 5.8
Erweitern Sie SIMPLE($\mathbb{Z}$) um Anweisungen der Form „**if** e **then** Π **fi**“ mit der üblichen Bedeutung von **if**-Anweisungen ohne **else**-Zweig.

- Geben Sie die Syntax und operationale Semantik der erweiterten Sprache an.
- Bestimmen Sie wp, wlp, sp für **if-then**-Anweisungen. Geben Sie Hoare-Regeln für partielle und totale Korrektheit an.

Behandeln Sie **if-then**-Anweisungen als Anweisungen erster Klasse, d.h., beziehen Sie sich bei der Festlegung der operationalen und axiomatischen Semantik nicht auf andere Anweisungentypen. (Bei der Herleitung dieser Semantik können diese aber verwendet werden, sofern sie danach eliminiert werden.)

Lösung 5.8
Syntax: Wir fügen den Produktionen, die die Syntax von SIMPLE($\mathbb{Z}$) beschreiben, eine weitere hinzu:

$$\textit{Programm} \to \text{„}\textbf{if}\text{“ } \textit{Term} \text{ „}\textbf{then}\text{“ } \textit{Programm} \text{ „}\textbf{fi}\text{“}$$

Die Semantik und die Verifikationsregeln lassen sich systematisch herleiten, indem man **if-then**-Anweisungen mittels $\textbf{if } e \textbf{ then } \Pi \textbf{ fi} \equiv \textbf{if } e \textbf{ then } \Pi \textbf{ else skip fi}$ in vollständige **if**-Anweisungen umschreibt. Die Semantik des Programms rechts ist bereits bekannt. Wir können sie bestimmen und vereinfachen, um die entsprechenden Regeln für das Programm links zu erhalten.

Strukturelle operationale Semantik:

$$(\textbf{if } e \textbf{ then } \Pi \textbf{ fi}, \sigma) = (\textbf{if } e \textbf{ then } \Pi \textbf{ else skip fi}, \sigma) \Rightarrow \begin{cases} (\Pi, \sigma) & \text{wenn } [e]\,\sigma \neq 0 \\ (\textbf{skip}, \sigma) \Rightarrow \sigma & \text{wenn } [e]\,\sigma = 0 \end{cases}$$

Insgesamt erhalten wir folgende SOS-Regel für die **if-then**-Anweisung:

$$(\textbf{if } e \textbf{ then } \Pi \textbf{ fi}, \sigma) \Rightarrow \begin{cases} (\Pi, \sigma) & \text{wenn } [e]\,\sigma \neq 0 \\ \sigma & \text{wenn } [e]\,\sigma = 0 \end{cases}$$

bzw. als Regeln für die induktive Definition von $\Rightarrow$ geschrieben:

$$\frac{[e]\,\sigma \neq 0}{(\textbf{if } e \textbf{ then } \Pi \textbf{ fi}, \sigma) \Rightarrow (\Pi, \sigma)} \qquad \frac{[e]\,\sigma = 0}{(\textbf{if } e \textbf{ then } \Pi \textbf{ fi}, \sigma) \Rightarrow \sigma}$$

Natürliche Semantik:

$$[\textbf{if } e \textbf{ then } \Pi \textbf{ fi}]\,\sigma = [\textbf{if } e \textbf{ then } \Pi \textbf{ else skip fi}]\,\sigma = \begin{cases} [\Pi]\,\sigma & \text{wenn } [e]\,\sigma \neq 0 \\ [\textbf{skip}]\,\sigma = \sigma & \text{wenn } [e]\,\sigma = 0 \end{cases}$$

Nach Streichen der Zwischenschritte in der Herleitung definieren wir die natürliche Semantik durch

$$[\textbf{if } e \textbf{ then } \Pi \textbf{ fi}]\,\sigma = \begin{cases} [\Pi]\,\sigma & \text{wenn } [e]\,\sigma \neq 0 \\ \sigma & \text{wenn } [e]\,\sigma = 0 \end{cases}$$

Weakest Precondition:

$$\begin{aligned} \mathsf{wp}(\textbf{if } e \textbf{ then } \Pi \textbf{ fi}, G) &= \mathsf{wp}(\textbf{if } e \textbf{ then } \Pi \textbf{ else skip fi}, G) \\ &= (e \wedge \mathsf{wp}(\Pi, G)) \vee (\neg e \wedge \mathsf{wp}(\textbf{skip}, G)) \\ &= (e \wedge \mathsf{wp}(\Pi, G)) \vee (\neg e \wedge G) \quad \text{oder} \quad (e \Rightarrow \mathsf{wp}(\Pi, G)) \wedge (\neg e \Rightarrow G) \end{aligned}$$

Weakest Liberal Precondition: analog wp

$$\mathsf{wlp}(\textbf{if } e \textbf{ then } \Pi \textbf{ fi}, G) = (e \wedge \mathsf{wlp}(\Pi, G)) \vee (\neg e \wedge G) \quad \text{oder} \quad (e \Rightarrow \mathsf{wlp}(\Pi, G)) \wedge (\neg e \Rightarrow G)$$

Strongest Postcondition:

$$\begin{aligned}\mathsf{sp}(F, \textbf{if } e \textbf{ then } \Pi \textbf{ fi}) &= \mathsf{sp}(F, \textbf{if } e \textbf{ then } \Pi \textbf{ else skip fi}) \\ &= \mathsf{sp}(F \wedge e, \Pi) \vee \mathsf{sp}(F \wedge \neg e, \textbf{skip}) \\ &= \mathsf{sp}(F \wedge e, \Pi) \vee (F \wedge \neg e)\end{aligned}$$

Hoare-Kalkül:

$$\dfrac{\dfrac{\{F \wedge e\}\,\Pi\,\{G\} \qquad \dfrac{(F \wedge \neg e) \Rightarrow G \quad \{G\}\,\textbf{skip}\,\{G\}}{\{F \wedge \neg e\}\,\textbf{skip}\,\{G\}}\ (\text{skip})}{\{F\}\,\textbf{if } e \textbf{ then } \Pi \textbf{ else skip fi}\,\{G\}}\ (\text{if})}{\{F\}\,\textbf{if } e \textbf{ then } \Pi \textbf{ fi}\,\{G\}}\ (\text{imp})\ (\text{umschreiben})$$

Reduzieren wir diese Ableitung auf die offenen Prämissen, erhalten wir die Regel

$$\dfrac{\{F \wedge e\}\,\Pi\,\{G\} \quad (F \wedge \neg e) \Rightarrow G}{\{F\}\,\textbf{if } e \textbf{ then } \Pi \textbf{ fi}\,\{G\}}$$