Karlsruher Institut für Technologie

Lehrstuhl für Programmierparadigmen
Sebastian Ullrich sebastian.ullrich@kit.edu
Max Wagner maximilian.wagner@kit.edu

Semantik von Programmiersprachen – SS 2019

http://pp.ipd.kit.edu/lehre/SS2019/semantik

Lösungen zu Blatt 9: Denotationale Semantik mit Fixpunktiteration Besprechung: 24.06.2019

1. Welche der folgenden Aussagen sind richtig, welche falsch? (H)

(a) Die folgende Rekursionsgleichung definiert die Funktion $f : \mathbb{N} \to \mathbb{N}$ eindeutig:

$$f(n) = \begin{cases} 2 \cdot f\left(\frac{n}{2}\right) & \text{falls } n \text{ gerade} \\ f(n+1) - 1 & \text{falls } n \text{ ungerade} \end{cases}$$

(b) Die folgende Rekursionsgleichung definiert die Funktion $g : \mathbb{N} \to \mathbb{N}$ eindeutig:

$$g(n) = \begin{cases} \left(g\left(\frac{n}{2}\right)\right)^2 & \text{falls } n \text{ gerade} \\ g(n-1) & \text{falls } n \text{ ungerade} \end{cases}$$

(c) Das Funktional $F : (\Sigma \rightharpoonup \Sigma) \to (\Sigma \rightharpoonup \Sigma)$ mit

$$F(f) = \lambda\sigma. \begin{cases} (f(\sigma[\mathtt{x} \mapsto 42]))[\mathtt{y} \mapsto 23] & \text{falls } \sigma(\mathtt{z}) \leq 17 \\ \sigma[\mathtt{z} \mapsto 0] & \text{sonst} \end{cases}$$

gehört zur Rekursionsgleichung

$$\mathcal{Q}(\sigma) = \begin{cases} (\mathcal{Q}(\sigma[\mathtt{x} \mapsto 42]))[\mathtt{y} \mapsto 23] & \text{falls } \sigma(\mathtt{z}) \leq 17 \\ \sigma[\mathtt{z} \mapsto 0] & \text{sonst} \end{cases}$$

(d) Für das Funktional $F : (\Sigma \rightharpoonup \Sigma) \to (\Sigma \rightharpoonup \Sigma)$ mit

$$F(f) = \lambda\sigma. \begin{cases} f(\sigma[\mathtt{y} \mapsto 12]) & \text{falls } \sigma(\mathtt{x}) = 13 \\ \sigma[\mathtt{x} \mapsto 10] & \text{sonst} \end{cases}$$

gilt

$$F^{42}(\bot)(\sigma) = \begin{cases} \bot & \text{falls } \sigma(\mathtt{x}) = 13 \\ \sigma[\mathtt{x} \mapsto 10] & \text{sonst} \end{cases}$$

(e) $\mathcal{D}[\![c_1\,;\ (c_2\,;\ c_3)]\!] = \mathcal{D}[\![(c_2\,;\ c_1)\,;\ c_3]\!]$

(f) $\mathcal{D}[\![\texttt{(if (x <= y) then z := x else z := y); skip}]\!]\,\sigma = \sigma[\mathtt{z} \mapsto \min(\sigma(\mathtt{x}),\ \sigma(\mathtt{y}))]$

Lösung:

(1a) Richtig. Man muss dafür zeigen, dass es eine Lösung gibt und dass die Rekursion irgendwann terminiert.

Die Identitätsfunktion id ist eine Lösung:

$$n = id(n) = \begin{cases} 2 \cdot \frac{n}{2} = 2 \cdot id\left(\frac{n}{2}\right) & \text{falls } n \text{ gerade} \\ (n+1) - 1 = id(n+1) - 1 & \text{falls } n \text{ ungerade} \end{cases}$$

Damit ist klar, dass die Rekursionsgleichung erfüllbar ist.

Für die Eindeutigkeit muss man nur noch beweisen, dass wenn f die Gleichung für alle n erfüllt, dann gilt $f(n) = n$ für alle n. Beweis per vollständiger Induktion über n:

Sei also n beliebig. Induktionsannahme: Für alle $m < n$ gilt: $f(m) = m$. Zu zeigen: $f(n) = n$.

- Fall $n = 0$: Aus der Rekursionsgleichung folgt: $f(0) = 2 \cdot f\left(\frac{0}{2}\right) = 2 \cdot f(0)$ und damit $f(0) = 0$.
- Fall $n = 1$: Dann gilt entsprechend: $f(1) = f(1+1) - 1 = 2 \cdot f(1) - 1$ und damit durch Umstellen $f(1) = 1$.
- Fall $n > 1$ gerade: Dann ist $\frac{n}{2} < n$, also folgt aus der Induktionsannahme mit $m = \frac{n}{2}$, dass $f\left(\frac{n}{2}\right) = \frac{n}{2}$. Damit gilt $f(n) = 2 \cdot f\left(\frac{n}{2}\right) = 2 \cdot \frac{n}{2} = n$.
- Fall $n > 1$ ungerade: Dann ist $\frac{n+1}{2} < n$, also folgt aus der Induktionsannahme mit $m = \frac{n+1}{2}$, dass $f\left(\frac{n+1}{2}\right) = \frac{n+1}{2}$. Damit gilt: $f(n) = f(n+1) - 1 = 2 \cdot f\left(\frac{n+1}{2}\right) - 1 = 2 \cdot \frac{n+1}{2} - 1 = n$.

Damit ist id die einzige Funktion, die die Rekursionsgleichung erfüllt.

(1b) Falsch. Die konstanten Funktionen $g_1(n) = 0$ und $g_2(n) = 1$ erfüllen beide die Rekursionsgleichung.

(1c) Richtig.

(1d) Richtig. Es gilt:

$$F^0(\bot)(\sigma) = \bot$$

$$F^1(\bot)(\sigma) = F(\bot)(\sigma) = \begin{cases} \bot & \text{falls } \sigma(\mathtt{x}) = 13 \\ \sigma[\mathtt{x} \mapsto 10] & \text{sonst} \end{cases}$$

$$F^2(\bot)(\sigma) = F(F^1(\bot))(\sigma) = \begin{cases} F^1(\bot)(\sigma[\mathtt{y} \mapsto 12]) = \bot & \text{falls } \sigma(\mathtt{x}) = 13 \\ \sigma[\mathtt{x} \mapsto 10] & \text{sonst} \end{cases} = F^1(\bot)$$

Entsprechend $F^{42}(\bot) = F^{41}(\bot) = \ldots = F^1(\bot)$

Intuitiv kann man sich als Begründung hierfür überlegen, wie ein Programm aussähe, dessen Semantik F entspricht.[1]. Der Schleifenrumpf in einem solchen Programm würde lediglich `y` verändern, aber als Schleifenbedingung lediglich den Wert von `x` benutzen. Dementsprechend kann die Semantik des rekursiven Falls (und alle Approximationen daran) nur $\bot$ sein, denn eine solche Schleife terminiert nie!

(1e) Falsch. Gegenbeispiel $c_1 =$ `x := 1`, $c_2 =$ `x := 0`, $c_3 =$ `skip`. Dann gilt:

$$\mathcal{D}[\![c_1\mathtt{;}\ (c_2\mathtt{;}\ c_3)]\!] = \mathcal{D}[\![c_2\mathtt{;}\ c_3]\!] \circ \mathcal{D}[\![c_1]\!] = (\mathcal{D}[\![c_3]\!] \circ \mathcal{D}[\![c_2]\!]) \circ \mathcal{D}[\![c_1]\!] = \lambda\sigma.\ \sigma[\mathtt{x} \mapsto 0]$$
$$\mathcal{D}[\![(c_2\mathtt{;}\ c_1)\mathtt{;}\ c_3]\!] = \mathcal{D}[\![c_3]\!] \circ \mathcal{D}[\![c_2\mathtt{;}\ c_1]\!] = \mathcal{D}[\![c_3]\!] \circ (\mathcal{D}[\![c_1]\!] \circ \mathcal{D}[\![c_2]\!]) = \lambda\sigma.\ \sigma[\mathtt{x} \mapsto 1]$$

Richtig wäre $\mathcal{D}[\![c_1\mathtt{;}\ (c_2\mathtt{;}\ c_3)]\!] = \mathcal{D}[\![(c_1\mathtt{;}\ c_2)\mathtt{;}\ c_3]\!]$. Zum Beweis einfach ausrechnen und die Assoziativität von $\circ$ ausnutzen.

[1] Rein formal betrachten wir diese Funktionale nur für While-Schleifen, deren Semantik bei Nichtausführung id ist, man muss also die Augen ein wenig zukneifen...

(1f) Richtig. Sei c das ganze Programm, *if* der if-Teil.

$$\begin{aligned}\mathcal{D}[\![P]\!]\sigma &= (\mathcal{D}[\![\texttt{skip}]\!] \circ \mathcal{D}[\![if]\!])(\sigma) = (id \circ \mathcal{D}[\![if]\!])(\sigma) = \mathcal{D}[\![if]\!]\sigma \\ &= \text{IF}\,(\mathcal{B}[\![\texttt{x <= y}]\!],\ \mathcal{D}[\![\texttt{z := x}]\!],\ \mathcal{D}[\![\texttt{z := y}]\!])\,\sigma \\ &= \text{IF}\,(\lambda\sigma.\ \sigma(\texttt{x}) \leq \sigma(\texttt{y}),\ \lambda\sigma.\ \sigma[\texttt{z} \mapsto \sigma(\texttt{y})],\ \lambda\sigma.\ \sigma[\texttt{z} \mapsto \sigma(\texttt{y})])\,\sigma \\ &= \begin{cases}\sigma[\texttt{z} \mapsto \sigma(\texttt{x})] & \text{falls } \sigma(\texttt{x}) \leq \sigma(\texttt{y}) \\ \sigma[\texttt{z} \mapsto \sigma(\texttt{y})] & \text{sonst}\end{cases} \\ &= \sigma[\texttt{z} \mapsto \min(\sigma(\texttt{x}),\ \sigma(\texttt{y}))]\end{aligned}$$

2. Fixpunktiteration (H)

Gegeben sei folgendes Programm P:

```
x := 0; i := n; while (1 <= i) do (x := x + 2 * i; i := i - 1)
```

(a) Geben Sie das Funktional F an, das in der denotationellen Semantik zur Schleife gehört.

(b) Berechnen Sie $F^0(\bot)$, $F^1(\bot)$, $F^2(\bot)$ und $F^3(\bot)$.

(c) Geben Sie $F^n(\bot)$ und (ein sinnvolles) $\text{FIX}\,(F)$ an (ohne Beweis).

(d) Geben Sie basierend darauf $\mathcal{D}[\![P]\!]$ an.

Lösung:

(2a) Zuerst die Semantik des Schleifenrumpfs:

$$\begin{aligned}\mathcal{D}[\![\texttt{x := x + 2 * i; i := i - 1}]\!]\sigma &= (\mathcal{D}[\![\texttt{i := i - 1}]\!] \circ \mathcal{D}[\![\texttt{x := x + 2 * i}]\!])(\sigma) \\ &= \mathcal{D}[\![\texttt{i := i - 1}]\!]\,(\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + 2 \cdot \sigma(\texttt{i})]) \\ &= \sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + 2 \cdot \sigma(\texttt{i}),\ \texttt{i} \mapsto \sigma(\texttt{i}) - 1]\end{aligned}$$

Damit ergibt sich das Funktional F zu:

$$\begin{aligned}F(f)(\sigma) &= \text{IF}\,(\mathcal{B}[\![\texttt{1 <= i}]\!],\ f \circ \mathcal{D}[\![\texttt{x := x + 2 * i; i := i - 1}]\!],\ id)\,\sigma \\ &= \text{IF}\,(\lambda\sigma.\ 1 \leq \sigma(\texttt{i}),\ \lambda\sigma.\ f(\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + 2 \cdot \sigma(\texttt{i}),\ \texttt{i} \mapsto \sigma(\texttt{i}) - 1]),\ id)\,\sigma \\ &= \begin{cases}\sigma & \text{falls } 1 > \sigma(\texttt{i}) \\ f(\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + 2 \cdot \sigma(\texttt{i}),\ \texttt{i} \mapsto \sigma(\texttt{i}) - 1]) & \text{falls } 1 \leq \sigma(\texttt{i})\end{cases}\end{aligned}$$

(2b) Sei *body* definiert als

$$\begin{aligned}body(\sigma) :&= \mathcal{D}[\![\texttt{x := x + 2 * i; i := i - 1}]\!]\sigma \\ &= \sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + 2 \cdot \sigma(\texttt{i}),\ \texttt{i} \mapsto \sigma(\texttt{i}) - 1]\end{aligned}$$

$$F^0(\bot)(\sigma) = \bot$$

$$F^1(\bot)(\sigma) = F(F^0(\bot))(\sigma) = \begin{cases} \sigma & \text{falls } 1 > \sigma(\mathtt{i}) \\ F^0(\bot)(\mathit{body}(\sigma)) = \bot & \text{falls } 1 \leq \sigma(\mathtt{i}) \end{cases}$$

$$F^2(\bot)(\sigma) = F(F^1(\bot))(\sigma) = \begin{cases} \sigma & \text{falls } 1 > \sigma(\mathtt{i}) \\ F^1(\bot)(\mathit{body}(\sigma)) & \text{falls } 1 \leq \sigma(\mathtt{i}) \end{cases}$$

$$= \begin{cases} \sigma & \text{falls } 1 > \sigma(\mathtt{i}) \\ \mathit{body}(\sigma) & \text{falls } 1 \leq \sigma(\mathtt{i}) \text{ und } 1 > \mathit{body}(\sigma)(\mathtt{i}) \\ \bot & \text{falls } 1 \leq \sigma(\mathtt{i}) \text{ und } 1 \leq \mathit{body}(\sigma)(\mathtt{i}) \end{cases}$$

$$= \begin{cases} \sigma & \text{falls } 1 > \sigma(\mathtt{i}) \\ \sigma[\mathtt{x} \mapsto \sigma(\mathtt{x}) + 2, \mathtt{i} \mapsto 0] & \text{falls } 1 = \sigma(\mathtt{i}) \\ \bot & \text{falls } 1 < \sigma(\mathtt{i}) \end{cases}$$

$$F^3(\bot)(\sigma) = F(F^2(\bot))(\sigma) = \begin{cases} \sigma & \text{falls } 1 > \sigma(\mathtt{i}) \\ F^2(\bot)(\mathit{body}(\sigma)) & \text{falls } 1 \leq \sigma(\mathtt{i}) \end{cases}$$

$$= \begin{cases} \sigma & \text{falls } 1 > \sigma(\mathtt{i}) \\ \mathit{body}(\sigma) & \text{falls } 1 \leq \sigma(\mathtt{i}) \text{ und } 1 > \mathit{body}(\sigma)(\mathtt{i}) \\ \mathit{body}(\sigma)[\mathtt{x} \mapsto \mathit{body}(\sigma)(\mathtt{x}) + 2, \mathtt{i} \mapsto 0] & \text{falls } 1 \leq \sigma(\mathtt{i}) \text{ und } 1 = \mathit{body}(\sigma)(\mathtt{i}) \\ \bot & \text{falls } 1 \leq \sigma(\mathtt{i}) \text{ und } 1 < \sigma'(\mathtt{i}) \end{cases}$$

$$= \begin{cases} \sigma & \text{falls } 1 > \sigma(\mathtt{i}) \\ \sigma[\mathtt{x} \mapsto \sigma(\mathtt{x}) + 2, \mathtt{i} \mapsto 0] & \text{falls } 1 = \sigma(\mathtt{i}) \\ \sigma[\mathtt{x} \mapsto \sigma(\mathtt{x}) + 6, \mathtt{i} \mapsto 0] & \text{falls } 2 = \sigma(\mathtt{i}) \\ \bot & \text{falls } 2 < \sigma(\mathtt{i}) \end{cases}$$

(2c)

$$F^n(\bot)(\sigma) = \begin{cases} \sigma & \text{falls } 1 > \sigma(\mathtt{i}) \\ \sigma[\mathtt{x} \mapsto \sigma(\mathtt{i}) \cdot (\sigma(\mathtt{i}) + 1) + \sigma(\mathtt{x}), \mathtt{i} \mapsto 0] & \text{falls } \sigma(\mathtt{i}) \in \{1, \ldots, n\} \\ \bot & \text{falls } n < \sigma(\mathtt{i}) \end{cases}$$

$$\mathrm{FIX}\,(F)\,\sigma = \begin{cases} \sigma & \text{falls } 1 > \sigma(\mathtt{i}) \\ \sigma[\mathtt{x} \mapsto \sigma(\mathtt{i}) \cdot (\sigma(\mathtt{i}) + 1) + \sigma(\mathtt{x}), \mathtt{i} \mapsto 0] & \text{falls } 1 \leq \sigma(\mathtt{i}) \end{cases}$$

(2d) Sei $w =$ `while (1 <= i) do (x := x + 2 * i; i := i - 1)`

$$\begin{aligned} \mathcal{D}[\![P]\!]\,\sigma &= (\mathcal{D}[\![w]\!] \circ \mathcal{D}[\![\mathtt{i := n}]\!] \circ \mathcal{D}[\![\mathtt{x := 0}]\!])(\sigma) = (\mathcal{D}[\![w]\!] \circ \mathcal{D}[\![\mathtt{i := n}]\!])(\sigma[\mathtt{x} \mapsto 0]) \\ &= \mathcal{D}[\![w]\!]\,(\sigma[\mathtt{x} \mapsto 0,\ \mathtt{i} \mapsto \sigma(\mathtt{n})]) = \mathrm{FIX}\,(F)\,(\sigma[\mathtt{x} \mapsto 0,\ \mathtt{i} \mapsto \sigma(\mathtt{n})]) \\ &= \begin{cases} \sigma[\mathtt{x} \mapsto 0,\ \mathtt{i} \mapsto \sigma(\mathtt{n})]) & \text{falls } \sigma(\mathtt{n}) < 1 \\ \sigma[\mathtt{x} \mapsto \sigma(\mathtt{n}) \cdot (\sigma(\mathtt{n}) + 1),\ \mathtt{i} \mapsto 0] & \text{falls } \sigma(\mathtt{n}) \geq 1 \end{cases} \end{aligned}$$

3. Lokale Variablen in Blöcken (Ü)

In Kap. 6.3 haben wir die While-Sprache um Blöcke mit lokalen Variablen erweitert und operationale Semantiken dafür angegeben. Erweitern Sie in diesem Abschnitt die denotationale Semantik entsprechend.

(a) Definieren Sie $\mathcal{D}[\![\cdot]\!]$ für das neue Sprachkonstrukt `{ var` x `=` a`; ` c `}`. Bleibt dadurch $\mathcal{D}[\![\cdot]\!]$ kompositional?

(b) Zeigen oder widerlegen Sie:

$$\mathcal{D}[\![\texttt{\{ var z = x; x := y; y := z \}}]\!] = \mathcal{D}[\![\texttt{x := x + y; y := x - y; x := x - y}]\!]$$

Lösung:

(3a) Letztendlich ist ein Block nichts anderes als die Zuweisung des Startwerts an x, der Ausführung von c und einer erneuten Zuweisung (des ursprünglichen Werts von x) an x. Damit ergibt sich:

$$\begin{aligned}\mathcal{D}[\![\texttt{\{ var } x \texttt{ = } a\texttt{; } c \texttt{ \}}]\!]\,\sigma &= ((\lambda\sigma'.\ \sigma'[x \mapsto \sigma(x)] \circ \mathcal{D}[\![c]\!] \circ (\lambda\sigma'.\ \sigma'[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]))(\sigma)\\ &= (\mathcal{D}[\![c]\!]\,(\sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]))[x \mapsto \sigma(x)]\end{aligned}$$

wobei das äußere Zustandsupdate Undefiniertheit propagieren soll. Die denotationale Semantik bleibt dadurch kompositional, da nur die Bedeutungen $\mathcal{A}[\![a]\!]$ und $\mathcal{D}[\![c]\!]$ der Teile a und c verwendet werden.

(3b) Beweis:

$$\begin{aligned}&\mathcal{D}[\![\texttt{\{ var z = x; x := y; y := z \}}]\!]\,\sigma\\ &= (\mathcal{D}[\![\texttt{x := y; y := z}]\!]\,(\sigma[\texttt{z} \mapsto \sigma(\texttt{x})]))[\texttt{z} \mapsto \sigma(\texttt{z})]\\ &= (\mathcal{D}[\![\texttt{y := z}]\!] \circ \mathcal{D}[\![\texttt{x := y}]\!])(\sigma[\texttt{z} \mapsto \sigma(\texttt{x})]))[\texttt{z} \mapsto \sigma(\texttt{z})]\\ &= (\mathcal{D}[\![\texttt{y := z}]\!]\,(\sigma[\texttt{z} \mapsto \sigma(\texttt{x}),\ \texttt{x} \mapsto (\sigma[\texttt{z} \mapsto \sigma(\texttt{x})])(\texttt{y})]))[\texttt{z} \mapsto \sigma(\texttt{z})]\\ &= (\mathcal{D}[\![\texttt{y := z}]\!]\,(\sigma[\texttt{z} \mapsto \sigma(\texttt{x}),\ \texttt{x} \mapsto \sigma(\texttt{y})]))[\texttt{z} \mapsto \sigma(\texttt{z})]\\ &= \sigma[\texttt{z} \mapsto \sigma(\texttt{x}),\ \texttt{x} \mapsto \sigma(\texttt{y}),\ \texttt{y} \mapsto (\sigma[\texttt{z} \mapsto \sigma(\texttt{x}),\ \texttt{x} \mapsto \sigma(\texttt{y})])(\texttt{z}),\ \texttt{z} \mapsto \sigma(\texttt{z})]\\ &= \sigma[\texttt{x} \mapsto \sigma(\texttt{y}),\ \texttt{y} \mapsto \sigma(\texttt{x}), \texttt{z} \mapsto \sigma(\texttt{z})] = \sigma[\texttt{x} \mapsto \sigma(\texttt{y}),\ \texttt{y} \mapsto \sigma(\texttt{x})]\end{aligned}$$

$$\begin{aligned}&\mathcal{D}[\![\texttt{x := x + y; y := x - y; x := x - y}]\!]\,\sigma\\ &= (\mathcal{D}[\![\texttt{x := x - y}]\!] \circ \mathcal{D}[\![\texttt{y := x - y}]\!] \circ \mathcal{D}[\![\texttt{x := x + y}]\!])\sigma\\ &= (\mathcal{D}[\![\texttt{x := x - y}]\!] \circ \mathcal{D}[\![\texttt{y := x - y}]\!])(\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + \sigma(\texttt{y})])\\ &= \mathcal{D}[\![\texttt{x := x - y}]\!]\,(\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + \sigma(\texttt{y}),\ \texttt{y} \mapsto (\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + \sigma(\texttt{y})])(\texttt{x}) - (\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + \sigma(\texttt{y})])(\texttt{y})])\\ &= \mathcal{D}[\![\texttt{x := x - y}]\!]\,(\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + \sigma(\texttt{y}),\ \texttt{y} \mapsto (\sigma(\texttt{x}) + \sigma(\texttt{y})) - \sigma(\texttt{y})])\\ &= \mathcal{D}[\![\texttt{x := x - y}]\!]\,(\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + \sigma(\texttt{y}),\ \texttt{y} \mapsto \sigma(\texttt{x})])\\ &= \sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + \sigma(\texttt{y}),\ \texttt{y} \mapsto \sigma(\texttt{x}),\\ &\qquad \texttt{x} \mapsto (\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + \sigma(\texttt{y}),\ \texttt{y} \mapsto \sigma(\texttt{x})])(\texttt{x}) - (\sigma[\texttt{x} \mapsto \sigma(\texttt{x}) + \sigma(\texttt{y}),\ \texttt{y} \mapsto \sigma(\texttt{x})])(\texttt{y})]\\ &= \sigma[\texttt{y} \mapsto \sigma(\texttt{x}),\ \texttt{x} \mapsto (\sigma(\texttt{x}) + \sigma(\texttt{y})) - \sigma(\texttt{x})] = \sigma[\texttt{y} \mapsto \sigma(\texttt{x}),\ \texttt{x} \mapsto \sigma(\texttt{y})]\end{aligned}$$