Prof. Dr. Ralf Hinze
Alexander Dinges, M.Sc.
Sebastian Schloßer, M.Sc.

RPTU
Fachbereich Informatik
AG Programmiersprachen

Lösungshinweise/-vorschläge zum Übungsblatt 2: Grundlagen der Programmierung (WS 2023/24)

Hinweis zu den Aufgabentypen:

- **Präsenzaufgaben** werden während der Übungsstunden gemeinsam in den Übungsgruppen zur Vorbereitung der Einreichaufgaben bearbeitet und besprochen. Es ist keine Abgabe erforderlich.
- **Einreichaufgaben** mit Punkten sind Pflichtaufgaben, die abgegeben werden müssen und für die Klausurzulassung relevant sind.
- **Trainingsaufgaben** werden nicht abgegeben, jedoch ist der dort behandelte Stoff durchaus klausurrelevant. Nutzen Sie diese Aufgaben zur Vertiefung des gelernten Stoffes und zur Klausurvorbereitung.

Sie können in den Sprech- und Übungsstunden zu allen Aufgabentypen Fragen stellen. Sie erhalten zu allen Aufgaben Lösungshinweise, jeweils nach der Abgabefrist für die letzte Übungsgruppe (Mittwoch Mittag).

Installationsanleitung Eine Installationsanleitung und Anweisungen zur Einrichtung der Entwicklungsumgebung finden Sie auf unserer Homepage.

Aufgabe 1 Statische und Dynamische Semantik (Präsenzaufgabe)

Motivation: Sie sollen anhand einiger Beispiele die bisher bekannten Regeln der statischen und dynamischen Semantik üben. Ein gutes Verständnis dieser Regeln und deren Anwendung in Beweisbäumen wird Ihnen im Laufe der Vorlesung an vielen Stellen helfen, sich Mini-F# systematisch zu erschließen. Sie können sich für diese Aufgabe an den Vorlesungsfolien 101 bis 126 sowie am Skript Kapitel 3.1 und 3.2 orientieren.

Prüfen Sie, ob die folgenden Mini-F#-Ausdrücke gültige Ausdrücke bezüglich der statischen Semantik sind. Geben Sie für die gültigen Ausdrücke deren Typ und einen entsprechenden Beweisbaum mit den Regeln der statischen Semantik an. Werten Sie dann gültige Ausdrücke mit den Regeln der dynamischen Semantik aus und geben Sie einen entsprechenden Beweisbaum an. Erklären Sie für die ungültigen Ausdrücke, wo der Fehler liegt.

Hier noch mal eine Übersicht der benötigten Regeln:

Syntax	**Statische Semantik**	**Dynamische Semantik**
n	$\dfrac{}{n : Nat}$	$\dfrac{}{n \Downarrow n}$
*	$\dfrac{e_1 : Nat \qquad e_2 : Nat}{e_1 * e_2 : Nat}$	$\dfrac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 * e_2 \Downarrow n_1 \cdot n_2}$
+	$\dfrac{e_1 : Nat \qquad e_2 : Nat}{e_1 + e_2 : Nat}$	$\dfrac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 + e_2 \Downarrow n_1 + n_2}$
false	$\dfrac{}{false : Bool}$	$\dfrac{}{false \Downarrow false}$
true	$\dfrac{}{true : Bool}$	$\dfrac{}{true \Downarrow true}$
if then else	$\dfrac{e_1 : Bool \qquad e_2 : t \qquad e_3 : t}{\texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 : t}$	$\dfrac{e_1 \Downarrow true \qquad e_2 \Downarrow v}{\texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 \Downarrow v}$ $\dfrac{e_1 \Downarrow false \qquad e_3 \Downarrow v}{\texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 \Downarrow v}$
<	$\dfrac{e_1 : Nat \qquad e_2 : Nat}{e_1 < e_2 : Bool}$	$\dfrac{e_1 \Downarrow n+k \qquad e_2 \Downarrow n}{e_1 < e_2 \Downarrow false}$ $\dfrac{e_1 \Downarrow n \qquad e_2 \Downarrow n+1+k}{e_1 < e_2 \Downarrow true}$
	Für andere Operationen genauso (z.B. <>)	

a) `(4 * 5) > (17 + 3)`

Statische Semantik

$$\dfrac{\dfrac{\dfrac{}{\texttt{4} : \mathit{Nat}} \quad \dfrac{}{\texttt{5} : \mathit{Nat}}}{\texttt{4 * 5} : \mathit{Nat}} \quad \dfrac{\dfrac{}{\texttt{17} : \mathit{Nat}} \quad \dfrac{}{\texttt{3} : \mathit{Nat}}}{\texttt{17 + 3} : \mathit{Nat}}}{\texttt{(4 * 5) > (17 + 3)} : \mathit{Bool}}$$

Dynamische Semantik

$$\dfrac{\dfrac{\dfrac{}{\texttt{4} \Downarrow 4} \quad \dfrac{}{\texttt{5} \Downarrow 5}}{\texttt{4 * 5} \Downarrow 20} \quad \dfrac{\dfrac{}{\texttt{17} \Downarrow 17} \quad \dfrac{}{\texttt{3} \Downarrow 3}}{\texttt{17 + 3} \Downarrow 20}}{\texttt{(4 * 5) > (17 + 3)} \Downarrow \mathit{false}}$$

b) `(5 > 4) > 3`

Ungültiger Ausdruck, da `5 > 4 : Bool` und `3 : Nat`.

c) **`if`** `(2 + 6) < 10` **`then`** `42` **`else`** `9`

Statische Semantik

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{2} : \mathit{Nat}} \quad \dfrac{}{\texttt{6} : \mathit{Nat}}}{\texttt{2 + 6} : \mathit{Nat}} \quad \dfrac{}{\texttt{10} : \mathit{Nat}}}{\texttt{(2 + 6) < 10} : \mathit{Bool}} \quad \dfrac{}{\texttt{42} : \mathit{Nat}} \quad \dfrac{}{\texttt{9} : \mathit{Nat}}}{\texttt{if (2 + 6) < 10 then 42 else 9} : \mathit{Nat}}$$

Dynamische Semantik

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{2} \Downarrow 2} \quad \dfrac{}{\texttt{6} \Downarrow 6}}{\texttt{2 + 6} \Downarrow 8} \quad \dfrac{}{\texttt{10} \Downarrow 10}}{\texttt{(2 + 6) < 10} \Downarrow \mathit{true}} \quad \dfrac{}{\texttt{42} \Downarrow 42}}{\texttt{if (2 + 6) < 10 then 42 else 9} \Downarrow 42}$$

d) **`if`** `42` **`then false else true`**

Ungültiger Ausdruck, da Bedingung nicht vom Typ *Bool*.

Aufgabe 2 Statische Semantik (Einreichaufgabe, 8 Punkte)

Motivation: Dies ist die Fortsetzung von Aufgabe 1. Hier betrachten wir ausschließlich die Statische Semantik, dieses Mal allerdings mit Wertedefinitionen. Sie können sich für diese Aufgabe an den Vorlesungsfolien 131 bis 141 sowie am Skript Kapitel 3.3 orientieren.

Prüfen Sie, ob die folgenden Mini-F#-Ausdrücke gültige Ausdrücke bezüglich der statischen Semantik sind. Geben Sie für die gültigen Ausdrücke deren Typ und einen entsprechenden Beweisbaum mit den Regeln der statischen Semantik an. Erklären Sie für die ungültigen Ausdrücke, wo der Fehler liegt.

Hinweis: Wir verwenden hier Wertedefinitionen. Sie müssen also, wie ab Vorlesungsfolie 138 gezeigt, entsprechend Signaturen angeben.

Prüfen Sie, ob die folgenden Ausdrücke typkorrekt sind. Falls ja, geben Sie einen vollständigen Beweisbaum an. Andernfalls begründen Sie kurz, warum der Ausdruck nicht typkorrekt ist.

a) `(if 1 > 0 then 3 else 4) + 1`

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \texttt{1} : \mathit{Nat}} \quad \dfrac{}{\emptyset \vdash \texttt{0} : \mathit{Nat}}}{\emptyset \vdash \texttt{1 > 0} : \mathit{Bool}} \quad \dfrac{}{\emptyset \vdash \texttt{3} : \mathit{Nat}} \quad \dfrac{}{\emptyset \vdash \texttt{4} : \mathit{Nat}}}{\emptyset \vdash \texttt{if (1 > 0) then 3 else 4} : \mathit{Nat}} \quad \dfrac{}{\emptyset \vdash \texttt{1} : \mathit{Nat}}}{\emptyset \vdash \texttt{(if (1 > 0) then 3 else 4) + 1} : \mathit{Nat}}$$

b) `let x = 815 in x + (x * x)`

$$\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \texttt{815} : \mathit{Nat}}}{\emptyset \vdash \texttt{let x = 815} : \{\texttt{x} \mapsto \mathit{Nat}\}} \quad \dfrac{\dfrac{}{\{\texttt{x} \mapsto \mathit{Nat}\} \vdash \texttt{x} : \mathit{Nat}} \quad \dfrac{\dfrac{}{\{\texttt{x} \mapsto \mathit{Nat}\} \vdash \texttt{x} : \mathit{Nat}} \quad \dfrac{}{\{\texttt{x} \mapsto \mathit{Nat}\} \vdash \texttt{x} : \mathit{Nat}}}{\{\texttt{x} \mapsto \mathit{Nat}\} \vdash \texttt{x * x} : \mathit{Nat}}}{\{\texttt{x} \mapsto \mathit{Nat}\} \vdash \texttt{x + (x * x)} : \mathit{Nat}}}{\emptyset \vdash \texttt{let x = 815 in x + (x * x)} : \mathit{Nat}}$$

c) `let x = false in if x then 1 < 2 else 2`

Ungültiger Ausdruck, da die Typen in den beiden Zweigen des `if`-Ausdrucks nicht übereinstimmen.

Aufgabe 3 Dynamische Semantik (Einreichaufgabe, 6 Punkte)

Motivation: Dies ist die Fortsetzung von Aufgabe 1. Hier betrachten wir ausschließlich die Dynamische Semantik, dieses Mal allerdings mit Wertedefinitionen. Sie können sich für diese Aufgabe an den Vorlesungsfolien 142 bis 147 sowie am Skript Kapitel 3.3 orientieren.

Werten Sie die Ausdrücke mit den Regeln der dynamischen Semantik aus und geben Sie einen entsprechenden Beweisbaum an.

Hinweis: Wir verwenden hier Wertedefinitionen. Sie müssen also, wie ab Vorlesungsfolie 143 gezeigt, entsprechend Umgebungen angeben.

a) `let x = false in (if x then 0 else 1)`

$$\dfrac{\dfrac{}{\emptyset \vdash \texttt{false} \Downarrow \mathit{false}}}{\dfrac{\emptyset \vdash \texttt{let x = false} \Downarrow \{\texttt{x} \mapsto \mathit{false}\}}{\emptyset \vdash \texttt{let x = false in (if x then 0 else 1)} \Downarrow 1}} \quad \dfrac{\dfrac{}{\{\texttt{x} \mapsto \mathit{false}\} \vdash \texttt{x} \Downarrow \mathit{false}} \quad \dfrac{}{\{\texttt{x} \mapsto \mathit{false}\} \vdash 1 \Downarrow 1}}{\{\texttt{x} \mapsto \mathit{false}\} \vdash \texttt{if x then 0 else 1} \Downarrow 1}$$

b) `let x = 1 in let x = 2 in x`

$$\dfrac{\dfrac{}{\emptyset \vdash 1 \Downarrow 1}}{\dfrac{\emptyset \vdash \texttt{let x = 1} \Downarrow \{\texttt{x} \mapsto 1\}}{\emptyset \vdash \texttt{let x = 1 in let x = 2 in x} \Downarrow 2}} \quad \dfrac{\dfrac{\dfrac{}{\{\texttt{x} \mapsto 1\} \vdash 2 \Downarrow 2}}{\{\texttt{x} \mapsto 1\} \vdash \texttt{let x = 2} \Downarrow \{\texttt{x} \mapsto 2\}} \quad \dfrac{}{\{\texttt{x} \mapsto 1, \texttt{x} \mapsto 2\} \vdash \texttt{x} \Downarrow 2}}{\{\texttt{x} \mapsto 1\} \vdash \texttt{let x = 2 in x} \Downarrow 2}$$

Aufgabe 4 Programmieren mit Zahlen (Einreichaufgabe, 6 Punkte)

Motivation: Dies ist unsere erste Programmieraufgabe. Sie sollen sich mit dem Setup vertraut machen und erste kleine Funktionen implementieren.

Hinweis: Arbeiten Sie die auf der Homepage verlinkte Installationsanleitung ab. Sie enthält wertvolle Hinweise zum Bearbeiten von Programmieraufgaben. Laden Sie sich die Vorlage von der Website herunter, öffnen Sie den Ordner wie beschrieben in VS Code und schreiben Sie Ihre Lösungen in die Datei `Zahlen.fs`. **Laden Sie abschließend ausschließlich diese Datei im ExClaim-System hoch.**

a) Definieren Sie eine Funktion `avg3 : Nat -> Nat -> Nat -> Nat`, die den (ganzzahligen) Durchschnitt dreier Zahlen berechnet.

Beispiele:

```
avg3 2N 5N 8N = 5N      avg3 5N 2N 3N = 3N      avg3 2N 5N 7N = 4N      avg3 13N 6N 0N = 6N
```

```
let avg3 (a: Nat) (b: Nat) (c: Nat): Nat =
    (a + b + c)/3N
```

b) Definieren Sie eine Funktion `min3 : Nat -> Nat -> Nat -> Nat`, die das Minimum dreier Zahlen berechnet.

Beispiele:

```
min3 2N 5N 8N = 2N      min3 5N 2N 3N = 2N      min3 12N 7N 5N = 5N      min3 13N 6N 0N = 0N
```

Tipp: Verwenden Sie die Funktion `min` von Vorlesungsfolie 162.

```
let min3 (a: Nat) (b: Nat) (c: Nat) =
    min (min a b) c
```

c) Definieren Sie eine Funktion `ceil10 : Nat -> Nat`, die eine Zahl auf die nächste durch 10 teilbare Zahl aufrundet.

Beispiele:

```
ceil10 2N = 10N         ceil10 10N = 10N        ceil10 12N = 20N        ceil10 29N = 30N
```

```
let ceil10 (a: Nat): Nat =
    if a % 10N = 0N then a else a + (10N - (a % 10N))
```