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

RPTU
Fachbereich Informatik
AG Programmiersprachen

Lösungshinweise/-vorschläge zum Übungsblatt 2: Grundlagen der Programmierung (WS 2024/25)

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 100 bis 125 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	$\dfrac{e_1 : Bool \qquad e_2 : t \qquad e_3 : t}{\texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 : t}$	$\dfrac{e_1 \Downarrow true \qquad e_2 \Downarrow v}{\texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 \Downarrow v}$ $\dfrac{e_1 \Downarrow false \qquad e_3 \Downarrow v}{\texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 \Downarrow v}$
<	$\dfrac{e_1 : Nat \qquad e_2 : Nat}{e_1 < e_2 : Bool}$ Für andere Operationen genauso (z.B. <>)	$\dfrac{e_1 \Downarrow n + k \qquad e_2 \Downarrow n}{e_1 < e_2 \Downarrow false}$ $\dfrac{e_1 \Downarrow n \qquad e_2 \Downarrow n + 1 + k}{e_1 < e_2 \Downarrow true}$

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 130 bis 140 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 137 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 <> 2 then 3 else 4) * 5`

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \texttt{1} : \text{Nat}} \quad \dfrac{}{\emptyset \vdash \texttt{2} : \text{Nat}}}{\emptyset \vdash \texttt{1 <> 2} : \text{Bool}} \quad \dfrac{}{\emptyset \vdash \texttt{3} : \text{Nat}} \quad \dfrac{}{\emptyset \vdash \texttt{4} : \text{Nat}}}{\emptyset \vdash \texttt{if 1 <> 2 then 3 else 4} : \text{Nat}} \quad \dfrac{}{\emptyset \vdash \texttt{5} : \text{Nat}}}{\emptyset \vdash \texttt{(if 1 <> 2 then 3 else 4) * 5} : \text{Nat}}$$

b) `let x = x in x`

Ungültiger Ausdruck, da das `x` auf der rechten Seite von `=` nicht definiert ist.

c) `let x = 815 in x = 816`

$$\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \texttt{815} : \text{Nat}}}{\emptyset \vdash \texttt{let x = 815} : \{x \mapsto \text{Nat}\}} \quad \dfrac{\dfrac{}{\{x \mapsto \text{Nat}\} \vdash \texttt{x} : \text{Nat}} \quad \dfrac{}{\{x \mapsto \text{Nat}\} \vdash \texttt{816} : \text{Nat}}}{\{x \mapsto \text{Nat}\} \vdash \texttt{x = 816} : \text{Bool}}}{\emptyset \vdash \texttt{let x = 815 in x = 816} : \text{Bool}}$$

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 141 bis 146 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 142 gezeigt, entsprechend Umgebungen angeben.

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

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

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

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

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 `inOrder : Nat -> Nat -> Nat -> Bool`, die zurückgibt, ob die 3 Argumente in zuhnemender Reinfolge sind.

Beispiele:

```
inOrder 2N 2N 8N =      inOrder 5N 2N 3N =      inOrder 2N 5N 7N =      inOrder 6N 13N 0N =
true                    false                   true                    false
```

```
let inOrder (a: Nat) (b: Nat) (c: Nat): Bool =
    a <= b && b <= c
```

b) Definieren Sie eine Funktion `median3 : Nat -> Nat -> Nat -> Nat`, die den Median dreier Zahlen berechnet. Verwenden Sie dazu die Vergleichsoperationen `min`, `max`, `<=`, etc.

Beispiele:

```
median3 2N 8N 5N = 5N   median3 5N 2N 3N = 3N   median3 12N 7N 5N = 7N   median3 13N 6N 6N = 6N
```

Tipp: Es gibt folgende 6 Möglichkeiten, wie 3 unterschiedliche Elemente angeordnet sein können. Wenn wir die Element in aufsteigender Reihenfolge $1, 2, 3$ nennen, so müssen wir für jede dieser möglichen Anordnungen das zweitgrößte Element 2 zurückgeben. Also z.B. im dritten Fall sollte `(median3) a` zurückgeben. Mit Vergleichsoperationen kann man den Suchraum sukzessive zwiespalten, bis man den zutreffenden Fall identifiziert hat.

a	b	c			
1	2	3	$b \leq c$		$a \leq b$
1	3	2	else		
2	3	1			
2	1	3	$b \leq c$		else
3	1	2			
3	2	1	else		

```
let median3 (a: Nat) (b: Nat) (c: Nat): Nat =
    if a <= b then
        if b <= c then b else max a c
    else
        if b >= c then b else min a c
```

c) Definieren Sie eine Funktion `takeMeTo : Bool -> (Nat -> Nat -> Nat)`, die für **`true`** die Funktion zurückgibt, die ihr erstes Argument zurückgibt, und für **`false`** die Funktion, die ihr zweites Argument zurückgibt.

Beispiele:

```
(takeMeTo true) 1N 2N  = 1N                    (takeMeTo false) 4N 3N = 3N
```

```
let takeMeTo (a: Bool): (Nat -> Nat -> Nat) =
    if a
      then (fun (x : Nat) (_ : Nat) -> x)
      else (fun (_ : Nat) (y : Nat) -> y)
```