Programmiersprachen und Übersetzer
Übung 8

Ausgabe am: **4. Juni 2024**
Abgabe bis: **11. Juni 2024**

Aufgabe 1 - Type Inference (4 Punkte)

(a) Bestimmen Sie, unter Benutzung der bekannten Typregeln für $\lambda_{\rightarrow}$ sowie des +-Operators,

$$\frac{(x : T) \in \Gamma}{\Gamma \vdash x : T} \text{ (T-Var)}$$

$$\frac{\Gamma, x : T \vdash t : T'}{\Gamma \vdash (\lambda x : T.t) : T \rightarrow T'} \text{ (T-Abs)}$$

$$\frac{\Gamma \vdash t_1 : T \rightarrow T' \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1\ t_2 : T'} \text{ (T-App)}$$

$$\frac{\Gamma \vdash t_1 : Int \qquad \Gamma \vdash t_2 : Int}{\Gamma \vdash t_1 + t_2 : Int} \text{ (T-Plus)}$$

den Typen des folgenden SML-Ausdrucks

```
fn x => fn y => y(x) + 1;
```

durch Aufstellen der *type constraints* und Lösen des Gleichungssystems per Unifikation.

Die Abgabe soll in einer allgemein lesbaren Datei `aufgabe1.*` erfolgen.

Aufgabe 2 - Kontrollflussgraphen (5 Punkte)

Gegeben ist der x86-Assembler Code in `aufgabe2.s`. Bilden Sie den funktionslokalen Kontrollflussgraphen (CFG) mit maximalen Basisblöcken.

fn u => fn y => y(u) + 1 ;

die obere Funktion ist ≡ λu.(λy.(y u + 1))

(λu.(λy((y u : α1 + 1) : α2)) : α3)) : α4

Constraint Generation:

GU

[[α4]] = [[u]] → [[α3]]
[[α3]] = [[y]] → [[α2]]
[[α2]] : int
[[α1]] : int
[[y]] = [[u]] → [[α1]]

Klammerung beachten! -0,5P.

Rules	Stack	Substitutions
	[[α4]] = [[u]] → [[α3]] [[α3]] = [[y]] → [[α2]] [[α2]] : int [[α1]] : int [[y]] = [[u]] → [[α1]]	
3	[[α3]] = [[y]] → [[α2]] [[α2]] : int [[α1]] : int [[y]] = [[u]] → [[α1]]	[[α4]] ↦ ([[u]] → [[α3]])
3	[[α2]] : int [[α1]] : int [[y]] = [[u]] → [[α1]]	[[α4]] ↦ ([[u]] → ([[y]] → [[α2]])) [[α3]] ↦ ([[y]] → [[α2]])
3	[[α1]] : int [[y]] = [[u]] → [[α1]]	[[α4]] ↦ ([[u]] → ([[y]] → ~~[[α2]]~~ int)) [[α3]] ↦ ([[y]] → ~~[[α2]]~~ int) [[α2]] : int
3	[[y]] = [[u]] → int	[[α4]] ↦ ([[u]] → ([[y]] → int)) [[α3]] ↦ ([[y]] → int) [[α2]] : int , [[α1]] : int

3

$[\![\alpha_4]\!] \mapsto ([\![\alpha_3]\!] \to ((([\![\alpha_2]\!] \to \text{int}) \to \text{int}))$

$[\![\alpha_3]\!] \mapsto ((([\![\alpha_2]\!] \to \text{int}) \to \text{int})$

$[\![\alpha_2]\!] : \text{int}$

$[\![\alpha_1]\!] : \text{int}$

$[\![y]\!] \mapsto ([\![\alpha_2]\!] \to \text{int})$ ✓

$[\![\alpha_4]\!] \to ((([\![\alpha_2]\!] \to \text{int}) \to \text{int})$ ✓

Die Abgabe soll in einer allgemein lesbaren Datei `aufgabe2.*` erfolgen.

Aufgabe 3 - Kurzschlussauswertung (3 Punkte)

Kurzschlussauswertung zeichnen sich dadurch aus, dass ihre Ausführung möglichst früh abbricht. Ein Beispiel in C ist der Ausdruck `a && b && c`.

Schreiben Sie ein Programm in der Intermediate Representation aus der Vorlesung, das das Verhalten des angefügten C-Codes imitiert. Sie müssen lediglich den Funktionskörper implementieren und können die Variablen a, b, c direkt benutzen.

```
int func(int a, int b, int c) {
   return a && b && c ? 1 : 0;
}
```

Die Abgabe soll in der Datei `aufgabe3.ir` geschehen.

Abgabe

Die Antworten müssen im bestehenden Repository **in einem neu zu erstellenden Verzeichnis "ex8"** eingereicht werden.