8. Übung (11.–15. Juni 2018)

Programmierung

Aufgabe 1

In der Vorlesung wurde das Funktionssymbol `s` eingeführt, um die natürlichen Zahlen aufzubauen. Gegeben ist das Prädikat `nat`, das genau alle natürlichen Zahlen enthält.

```
1 nat(0).
2 nat(s(X)) :- nat(X).
```

(a) Geben Sie eine dreistellige Relation `div` an, die für jedes Paar von natürlichen Zahlen n und m, wobei $m \neq 0$, das Tripel $(n, m, \lfloor \frac{n}{m} \rfloor)$ enthält und sonst nichts. Nutzen Sie dafür die dreistellige Relation `sum` aus der Vorlesung.

Hinweis: Definieren Sie zunächst eine zweistellige Relation `lt`, die $<$ modelliert.

(b) Im folgenden kürzen wir mit `<n>` die natürliche Zahl `n` ab. Geben Sie eine SDL-Refutation für `?- div(<3>, <2>, <1>).` an.

Aufgabe 2 (AGS 14.1 ⋆)

Gegeben sei folgendes C_0-Programm *Max*:

```
1  #include <stdio.h>
2
3  int main() {
4    int a, b, max;
5    scanf("%i", &a);
6    scanf("%i", &b);
7    if (a > b) max = a;
8    else max = b;
9    printf("%d", max);
10   return 0;
11 }
```

(a) Berechnen Sie schrittweise das baumstrukturierte Programm $bMax_0 = \text{trans}(Max)$ mit Hilfe der in der Vorlesung angegebenen Übersetzungsfunktionen.

(b) Wandeln Sie $bMax_0$ in ein Programm Max_0 mit linearisierten Adressen um und berechnen Sie $\mathcal{P}[\![Max_0]\!](5 : 7)$. Dokumentieren Sie den Zustand der AM_0 wie üblich.

Aufgabe 3 (AGS 14.10)

(a) Geben Sie für folgendes C_0-Programm die Übersetzung in ein linearisiertes AM_0-Programm an. Zwischenschritte der Übersetzung brauchen Sie nicht anzugeben.

```
1  #include <stdio.h>
2
3  int main() {
4    int x, y, a;
5    scanf("%i", &y);
6    scanf("%i", &a);
7    x = 0;
8    while (x < a) {
9      x = x + 1;
10     y = y * y;
11   }
12   printf("%d", y);
13   return 0;
14 }
```

(b) Folgendes AM_0-Programm sei gegeben:

```
1: READ 1;          4: LOAD 2;          7: JMC 9;
2: READ 2;          5: LIT  0;          8: JMP 5;
3: LOAD 1;          6: SUB;             9: WRITE 2;
```

Protokollieren Sie den schrittweisen Ablauf dieses Programms auf der AM_0 mit der Anfangskonfiguration $(1, \varepsilon, [], 0 : 1, \varepsilon)$.

Aufgabe 4 (AGS 15.18 b)

Gegeben sei folgender AM_1-Code:

```
1:  INIT 1;            8:  LOADI(-2);          15: LOADA(global, 1);
2:  CALL 13;           9:  LIT 2;              16: PUSH;
3:  INIT 0;            10: DIV;                17: CALL 3;
4:  LOADI(-2);         11: STOREI(-2);         18: WRITE(global, 1);
5:  LIT 2;             12: RET 1;              19: JMP 0;
6:  GT;                13: INIT 0;
7:  JMC 12;            14: READ(global, 1);
```

Erstellen Sie ein Ablaufprotokoll der AM_1, indem Sie sie schrittweise ablaufen lassen, bis die Maschine terminiert. Sie Anfangskonfiguration sei $(14, \varepsilon, 0 : 0 : 1, 3, 4, \varepsilon)$. Sie müssen nur Zellen ausfüllen, deren Wert sich im Vergleich zur letzten Zeile geändert hat.

Zusatzaufgabe 1 (AGS 14.14)

(a) Gegeben sei folgendes C_0-Programm.

```
1  #include <stdio.h>
2  int main()
3  {
4    int x1, x2;
5    scanf("%i", &x1);
6    scanf("%i", &x2);
7    while (x1 > 0){
8      x1 = x2 - x1;
9      if (x2 > x1)
10       x2 = x2 / 2;
11   }
12   printf("%d", x1);
13   return 0;
14 }
```

Übersetzen Sie das Programm mittels *trans* in AM_0-Code mit linearen Adressen. Geben Sie nur das Endergebnis der Übersetzung, keine Zwischenschritte an!

(b) Gegeben sei der folgende Ausschnitt aus einem AM_0-Programm.

```
3:  LOAD 2;        6:  JMC 14;        9:  LIT 2;        12: STORE 2;
4:  LIT 5;         7:  LOAD 1;        10: MUL;          13: JMP 3;
5:  LT;            8:  LOAD 2;        11: ADD;          14: WRITE 1;
```

Erstellen Sie ein Ablaufsprotokoll für dieses Programmfragment, bis die AM_0 terminiert. Die Startkonfiguration ist $(7, \varepsilon, [1/3, 2/1], \varepsilon, \varepsilon)$.