10. Übung (17.–21. Juni 2018)

Programmierung

Beachten Sie die Übungsverlegungen zum 18.–20. Juni.

Aufgabe 1 (AGS 14.1 ⋆)

Gegeben sei folgendes C_0-Programm *Max*:

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

(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 in einer Tabelle.

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

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

Zusatzaufgabe 1 (AGS 13.7)

Gegeben sei das dreistellige Prädikat `append`, welches die Verkettung von Listen ausdrückt.

```
append([], Ys, Ys).
append([X|Xs], Ys, [X|Zs]) :- append(Xs, Ys, Zs).
```

(a) Geben Sie unter Nutzung von `append` ein Prologprogramm für das zweistellige Prädikat `reverse` an, welches erfüllt ist, falls das erste Argument eine Liste `Xs` ist und das zweite Argument `Ys` die Elemente von `Xs` in umgekehrter Reihenfolge enthält. Beispielsweise soll gelten: `reverse([1,2,3], [3,2,1]).`

(b) Eine Liste wird *Palindrom* genannt, falls die Folgen ihrer Elemente von vorn nach hinten und von hinten nach vorn gleich sind. Die Liste `[1,2,3,2,1]` ist zum Beispiel ein Palindrom. Bilden Sie diese Eigenschaft als einstelliges Prädikat `palindrom` in Prolog ab. Sie dürfen die Prädikate `append` und `reverse` benutzen.

(c) Gegeben sei das folgende Prolog$^-$-Programm:

```
1 s(alice, bob).
2 s(bob, charly).
3 s(alice, dory).
4 s(eve, alice).
5
6 t(X, Y) :- s(X, Y).
7 t(X, Z) :- s(X, Y), t(Y, Z).
```

Geben Sie alle SLD-Refutationen für `?- t(alice, X).` an. Geben Sie dabei die Belegungen für alle Variablen an. Geben Sie die Menge der möglichen Belegungen für `X` an.

Zusatzaufgabe 2 (AGS 14.14)

(a) Gegeben sei folgendes C_0-Programm.

```
1  #include <stdio.h>
2
3  int main() {
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 $(7, \varepsilon, [1/3, 2/1], \varepsilon, \varepsilon)$.