Technische Universität München
Prof. J. Esparza / Dr. M. Luttenberger, S. Sickert

Sommer 2016
11. Juli 2016

HA-Lösung

TA-Lösung

Einführung in die theoretische Informatik – Aufgabenblatt 11

Beachten Sie: Soweit nicht explizit angegeben, sind Ergebnisse stets zu begründen!

Hausaufgaben: Abgabe bis zum 06.07.2016 (Mittwoch) um 12:00

Aufgabe 11.1 — 2P+1P

Wir übersetzen primitiv rekursive und μ-rekursive Funktionen in Programme der Form definiert in TA10.3. Geben Sie hierzu für jedes Konstrukt der PR- und μ-rekursiven Funktionen den entsprechend erzeugten Code an.

(a) Geben Sie die Regeln für primitiv rekursive Funktionen an.

(b) Erweitern Sie (a) und geben Sie die Regeln für μ-rekursive Funktionen an.

Lösung:

Basisfunktionen $0, s, \pi_i^k$:

```
ℕ const_0() {
  return 0;
}

ℕ s(ℕ x) {
  return x + 1;
}

ℕ pi_k_i(ℕ x_1, ℕ x_2, ..., ℕ x_k) {
  return x_i;
}
```

Komposition. Sei $f(x_1, \ldots, x_j) = g(h_1(x_1, \ldots x_j), h_2(x_1, \ldots x_j), \ldots h_k(x_1, \ldots x_j))$

```
ℕ compose_g_h(ℕ x_1, ℕ x_2, ..., ℕ x_j) {
  return g(h_1(x_1, ... x_j), h_2(x_1, ... x_j), ..., h_k(x_1, ... x_j));
}
```

Rekursion:

```
ℕ recursion(ℕ x_1, ℕ x_2, ..., ℕ x_k) {
  if (x_1 == 0) {
    return g(x_2, .... x_k)
  }

  ℕ m = x_1 - 1;
  return h(f(m, x_2, ... x_k), m, x_2, ... x_k));
}
```

μ-Rekursion

```
ℕ mu_recursion_f(ℕ x_1, ℕ x_2, ..., ℕ x_k) {
  return mu_recursion_counter(0, x_1, x_2, ... x_k));
}

ℕ mu_recursion_f_find(ℕ n, ℕ x_1, ℕ x_2, ..., ℕ x_k) {
  ℕ y = f(n, x_1, ... x_k);

  if (y == 0) {
    return n;
  }
```

```
  N m = n + 1;
  return mu_recursion_f_find(m, x_1, x_2, ... x_k));
}
```

Aufgabe 11.2 **3P+2P**

Sei P das folgende folgendes GOTO-Programm mit Stack mit Eingabevariablen x_1, x_2 und Ausgabevariable x_0 und $f_P(x_1, x_2)$ die von P berechnete Funktion:

```
0       PUSH x_1;
1       PUSH x_2;
2       x_4 := 1;
3   M_0: x_2 := POP;
4       x_3 := EMPTY;
5       IF x_3 = 1 GOTO M_5;
6       x_1 := POP;
7       IF x_1 = 0 GOTO M_1;
8       GOTO M_2;
9   M_1: x_2 := x_2 + 1;
10      PUSH x_2;
11      GOTO M_0;
12  M_2: x_5 := x_1 - 1;
13      IF x_2 = 0 GOTO M_3;
14      GOTO M_4;
15  M_3: PUSH x_5;
16      PUSH x_4;
17      GOTO M_0;
18  M_4: PUSH x_5;
19      PUSH x_1;
20      x_5 := x_2 - 1;
21      PUSH x_5;
22      GOTO M_0;
23  M_5: x_0 := x_2;
24      HALT;
```

(a) Beschreiben Sie den Ablauf des Programms auf Eingabe $x_1 = 1, x_2 = 0$ explizit.

Stellen Sie eine Konfiguration in der Form $(l, x_0, x_1, x_2, x_3, x_4, x_5, s)$ dar, wobei s der Stackinhalt ist und l die Zeilennummer der Anweisungm, die als nächstes ausgeführt werden wird. Beschreiben Sie diesen in der Form $\langle s_1, s_2, \ldots, s_k \rangle$, wobei s_1 die unterste Zahl auf dem Stack ist, s_k die oberste Zahle.

Die ersten Schritte des Programms bei Aufruf mit $x_1 = 1, x_2 = 0$ sind hiermit:

$$(0,0,1,0,0,0,0,\langle\rangle) \to (1,0,1,0,0,0,0,\langle 1\rangle) \to (2,0,1,0,0,0,0,\langle 1,0\rangle) \to (3,0,1,0,0,1,0,\langle 1,0\rangle) \to (4,0,1,0,0,1,0,\langle 1\rangle)$$

Beschreiben Sie entsprechend den Rest des Programmablaufs.

(b) Übersetzen Sie P zunächst in ein WHILE-Programm P' mit Stack, d.h. neben den Anweisung für ein WHILE-Programm dürfen auch die Anweisungen **PUSH**,**POP**,**EMPTY** verwendet werden, die sich wie im Fall eines GOTO-Programms mit Stack verhalten.

Übersetzen Sie P' dann in ein Programm P'' der Form3, welche in TA10.3 definiert wurde (C-ähnliche Prozeduren mit $\mathbb{N}$ als einzigem Datentyp, ohne Schleifen).

Argumentieren Sie anhand von P'', welche Funktion $f_P(x_1, x_2)$ durch P berechnet wird.

Lösung:

(a) 0: $(0,0,1,0,0,0,0,\langle\rangle)$ – Instruction: PSH x1

1: $(1,0,1,0,0,0,0,\langle 1\rangle)$ – Instruction: PSH x2

2: $(2,0,1,0,0,0,0,\langle 1,0\rangle)$ – Instruction: SET x4 1

3: $(3,0,1,0,0,1,0,\langle 1,0\rangle)$ – Instruction: M0 POP x2

4: $(4,0,1,0,0,1,0,\langle 1\rangle)$ – Instruction: EMP x3

5: $(5,0,1,0,0,1,0,\langle 1\rangle)$ – Instruction: JEQ x3 1 M5

6: $(6,0,1,0,0,1,0,\langle 1\rangle)$ – Instruction: POP x1

7: $(7,0,1,0,0,1,0,\langle\rangle)$ – Instruction: JEQ x1 0 M1

8: $(8,0,1,0,0,1,0,\langle\rangle)$ – Instruction: JMP M2

9: $(12,0,1,0,0,1,0,\langle\rangle)$ – Instruction: M2 ADD x5 x1 -1

10: $(13,0,1,0,0,1,0,\langle\rangle)$ – Instruction: JEQ x2 0 M3

11: $(15, 0, 1, 0, 0, 1, 0, \langle\rangle)$ – Instruction: M3 PSH x5

12: $(16, 0, 1, 0, 0, 1, 0, \langle 0\rangle)$ – Instruction: PSH x4

13: $(17, 0, 1, 0, 0, 1, 0, \langle 0, 1\rangle)$ – Instruction: JMP M0

14: $(3, 0, 1, 0, 0, 1, 0, \langle 0, 1\rangle)$ – Instruction: M0 POP x2

15: $(4, 0, 1, 1, 0, 1, 0, \langle 0\rangle)$ – Instruction: EMP x3

16: $(5, 0, 1, 1, 0, 1, 0, \langle 0\rangle)$ – Instruction: JEQ x3 1 M5

17: $(6, 0, 1, 1, 0, 1, 0, \langle 0\rangle)$ – Instruction: POP x1

18: $(7, 0, 0, 1, 0, 1, 0, \langle\rangle)$ – Instruction: JEQ x1 0 M1

19: $(9, 0, 0, 1, 0, 1, 0, \langle\rangle)$ – Instruction: M1 ADD x2 x2 1

20: $(10, 0, 0, 2, 0, 1, 0, \langle\rangle)$ – Instruction: PSH x2

21: $(11, 0, 0, 2, 0, 1, 0, \langle 2\rangle)$ – Instruction: JMP M0

22: $(3, 0, 0, 2, 0, 1, 0, \langle 2\rangle)$ – Instruction: M0 POP x2

23: $(4, 0, 0, 2, 0, 1, 0, \langle\rangle)$ – Instruction: EMP x3

24: $(5, 0, 0, 2, 1, 1, 0, \langle\rangle)$ – Instruction: JEQ x3 1 M5

25: $(23, 0, 0, 2, 1, 1, 0, \langle\rangle)$ – Instruction: M5 ADD x0 x2 0

26: $(24, 2, 0, 2, 1, 1, 0, \langle\rangle)$ – Instruction: HLT

(b) WHILE mit Stack:

```
PUSH x1;
PUSH x2;
x4 := 1;
x6 := 1;
WHILE x6 ≠ 0 DO
  x2 := POP;
  x3 := EMPTY;
  IF x3 = 1 THEN
    x0 := x2;
    x6 := 0;
  ELSE
    x1 := POP;
    IF x1 = 0 THEN
      x2 := x2 + 1;
      PUSH x2;
    ELSE
      x5 := x1 − 1;
      IF x2 = 0 THEN
        PUSH x5;
        PUSH x4;
      ELSE
        PUSH x5;
        PUSH x1;
        x5 := x2 − 1;
        PUSH x5;
      END;
    END;
  END;
END;
```

Die While-Schleife muss nun in eine Rekursion übersetzt werden. Im Schleifenrumpf werden dabei maximal 2 Werte vom Stack konsumiert, wobei das Programm hält, sollte auf den Stack nur noch ein Wert liegen, welcher in diesem Fall der Rückgabewert auch ist. Insofern sollte sich der Rumpf in eine Funktion mit zwei Eingabeparametern übersetzen lassen. Schiebt man mindestens zwei Werte auf den Stack, so kann direkt rekursiv in den Schleifenrumpf gesprungen werden. Schiebt man nur einen Wert auf den Stack, so muss dieser rekursiv an die aufrufende Instanz zurückgegeben werden. Damit:

```
ℕ a(ℕ x1, ℕ x2) {
  if(x1 == 0) { //M1: a(0,x2) = x2+1
    x2 += 1;
    return x2;
  } else { //M2
    if(x2 == 0) { //M3: x1 > 0 && a(x1,0) = a(x1-1,1)
      x1 -= 1;
      return a(x1,1);
    } else { //M4: x1 >0 && x2 > 0 && a(x1,x2) = a(x1-1,a(x1,x2-1))
      x2 -= 1;
      N temp = a(x1,x2);
      x1 -= 1;
```

```
        return a(x1,temp);
      }
    }
  }

  N main(N x1, N x2) {
    N x0 = 0;

    return a(x1,x2);
  }
```

Es folgt, dass die Ackermann-Funktion berechnet wird.

Aufgabe 11.3 **2P+2P+3P**

Sei $a(k,n)$ die Ackermann-Funktion. Für festes $k \in \mathbb{N}$ sei $a_k(n) := a(k,n)$.

(a) Geben Sie ein LOOP-Programm an, das $a_2(n)$ berechnet.

(b) Geben Sie ein LOOP-Programm an, das $a_3(n)$ berechnet.

(c) Zeigen Sie, dass zu jedem $k \in \mathbb{N}$ ein LOOP-Programm P_k existiert, das die Funktion $a_k(n)$ berechnet.

(a) Wir verwenden die Gleichung $a(2,n) = 2(n+3) - 3$ aus den Folien um $a_2(x_1)$ zu definieren:

```
x1 := x1 + 3;
x2 := 2;
x0 := x1 * x2;
x0 := x0 - 3;
```

(b) Da $a_2(x)$ LOOP-berechenbar ist, können wir a_3 wie folgt definieren:

```
x0 := a2(1);
LOOP x1 DO
  x0 := a2(x0);
END;
```

(c) Wir folgern induktiv für alle $k \in \mathbb{N}$, dass a_k primitiv rekursiv ist. Die Basisfälle $k = 0$ ist trivial. Für a_{k+1} lässt sich dann folgendes LOOP-Programm P_{k+1} angeben:

```
x0 := Pk(1);
LOOP x1 DO
  x0 := Pk(x0);
END;
```

Aufgabe 11.4 **3P+2P**

Zeigen Sie jeweils explizit, dass die folgenden Funktionen primitiv rekursiv sind.

(a) $div(x,y) = x \text{ div } y$

(b) $mod(x,y) = x \text{ mod } y$

Verwenden Sie für die Definition der Funktionen nur die Basisfunktionen und das erweiterte Schema der primitiven Rekursion. Weiterhin dürfen Sie verwenden, dass die Addition, Multiplikation, die modifizierte Differenz und der beschränkten *max*-Operator primitiv rekursiv sind. Für alle andere Funktionen müssen Sie diese Eigenschaft zeigen.

Der Beweis muss direkt sein! Die Verwendung von LOOP-Programmen ist nicht gestattet.

Lösung:

(a)

$$lesseq(x,y) = 1 \dot{-} (x \dot{-} y)$$
$$div(x,y) = max\{z \leq x \mid lesseq(z * y, x)\}$$

(b)

$$mod(x,y) = x \dot{-} (y * (div(x,y))$$

Tutoraufgaben: Besprechung in KW27

Hinweis: Verwenden Sie für die Definition der folgenden Funktionen nur die Basisfunktionen und das erweiterte Schema der primitiven Rekursion. Weiterhin dürfen Sie verwenden, dass die Addition, Multiplikation, die modifizierte Differenz und der beschränkten *max*-Operator primitiv rekursiv sind. Für alle andere Funktionen müssen Sie diese Eigenschaft zeigen.

Der Beweis muss direkt sein! Die Verwendung von LOOP-Programmen ist nicht gestattet.

Aufgabe 11.1

Zeigen Sie, dass folgende Funktionen primitiv rekursiv sind:

(a)
$$\mathsf{ite}(x_0, x_1, x_2) = \begin{cases} x_1 & \text{falls } x_0 \neq 0 \\ x_2 & \text{sonst} \end{cases}$$

(b) Sei F_n die n-te Fibonacci-Zahl (mit $F_0 = 0$, $F_1 = 1$, $F_{n+2} = F_n + F_{n+1}$). Sei dann $f(n)$:
$$f(n) := F_n$$

Hinweis: Bei dieser Aufgabe dürfen Sie verwenden, dass die Cantorsche Paarungsfunktion c und die Umkehrungsfunktionen p_1, p_2 primitiv rekursiv sind.

Definieren Sie die Funktionen jeweils einmal nur unter Verwendung des normalen Schemas der primitiven Rekursion und zum Vergleich einmal unter Verwendung des erweiterten Schemas.

Lösung:

(a)
- Direkt mit Schema der primitiven Rekursion:

  $g\colon \mathbb{N}^2 \to \mathbb{N}\colon (x_1, x_2) \mapsto x_2$ oder kurz $g(x_1, x_2) := x_2$.

  $h\colon \mathbb{N}^4 \to \mathbb{N}\colon (z, m, x_1, x_2) \mapsto x_1$ oder kurz $h(z, m, x_1, x_2) := x_1$.

  Damit:
  $$\mathsf{ite}\colon \mathbb{N}^3 \to \mathbb{N}\colon (x_0, x_1, x_2) \mapsto \begin{cases} g(x_1, x_2) & \text{falls } x = 0 \\ h(f(x-1), x-1, x_1, x_2) & \text{sonst} \end{cases}$$

- Modular und mit dem erweiterten Schema:
  $$\mathsf{iszero}(0) = s(0) \qquad \mathsf{iszero}(m+1) = 0$$
  $$\mathsf{not}(x) = 1 \dot{-} x$$

  Damit:
  $$\mathsf{ite}(x_0, x_1, x_2) = (\mathsf{iszero}(x_0) * x_2) + (\mathsf{not}(\mathsf{iszero}(x_0)) * x_1)$$

(b) Mit dem erweiterten Schema der PR und der Cantorschen Paarungsfunktion c samt Umkehrfunktionen p_1, p_2:
$$\hat{f}(0) := c(0, s(0)) \qquad \hat{f}(n+1) := c(p_2(\hat{f}(n)), p_1(\hat{f}(n)) + p_2(\hat{f}(n))) \qquad f(n) := p_1(\hat{f}(n))$$

Mit dem normalen Schema:

$g\colon \mathbb{N}^0 \to \mathbb{N}\colon () \mapsto c(0, s(0))$

$h\colon \mathbb{N}^2 \to \mathbb{N}\colon (x, y) \mapsto c(p_2(x), p_1(x) + p_2(x))$

Damit:
$$\hat{f}\colon \mathbb{N} \to \mathbb{N}\colon x \mapsto \begin{cases} g() & \text{falls } x = 0 \\ h(\hat{f}(x-1), x-1) & \text{sonst} \end{cases}$$

und wieder $f(n) := p_1(\hat{f}(n))$.

Aufgabe 11.2

Sei $P(x)$ mit $x \in \mathbb{N}$ ein primitiv rekursives Prädikat. Zeigen Sie, dass die Funktionen

(a) $f(n) := \forall x \leq n.\ P(x)$ und

(b) $g(n) := \min\{x \leq n \mid P(x)\}$ (wobei $min\ \emptyset = n + 1$)

primitiv rekursiv sind.

Lösung:

(a)

$$\begin{aligned} f(0) &= P(0) \\ f(n+1) &= P(n+1) * f(n) \end{aligned}$$

(b)

$$\begin{aligned} eq(x,y) &= 1 \dot{-} ((x \dot{-} y) + (y \dot{-} x)) \\ g(0) &= ite(P(0), 0, 1) \\ g(n+1) &= ite(eq(g(n), n+1), ite(P(n+1), n+1, n+2), g(n)) \end{aligned}$$

Aufgabe 11.3

Sei $a : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ die Ackermann-Funktion. Zeigen Sie, dass $f(m,n) := \log_2(a(m,n))$ nicht primitiv rekursiv ist, wobei $\log_2$ der abgerundete ganzzahlige Logarithmus zur Basis 2 ist (mit $\log_2 0 := 0$).

Lösung: Angenommen, f sei primitiv rekursiv. Dann ist auch $h(m,n) := 2^{f(m,n)+1}$ primitiv rekursiv, da

$$\begin{aligned} twopow(0) &= 1 \\ twopow(n+1) &= 2 * twopow(n) \end{aligned}$$

Somit gilt aber $a(m,n) \leq h(m,n)$. Widerspruch zu Lemma 4.49!