Übung zur Vorlesung Berechenbarkeit und Komplexität

Aufgabe T15

Entwickeln Sie ein WHILE-Programm, daß den Wert 2^{x_1} berechnet. Analysieren Sie die Laufzeit ihres Programmes im uniformen und im logarithmischen Kostenmaß.

Aufgabe T16

Die Ackermannfunktion $A : \mathbb{N}^2 \to \mathbb{N}$ wurde in der Vorlesung folgendermaßen definert:

$$\begin{array}{lcll} A(0,m) & = & m+1 & \text{für } m \geq 0 \\ A(n+1,0) & = & A(n,1) & \text{für } n \geq 0 \\ A(n+1,m+1) & = & A(n,A(n+1,m)) & \text{für } n,m \geq 0 \end{array}$$

a) Zeigen Sie, daß die Ackermannfunktion für alle Parameter $n, m \in \mathbb{N}$ terminiert.

b) Beweisen Sie durch Induktion nach n folgende Aussage:

$$A(n,m) \leq A(n+1,m-1) \text{ für alle } m \geq 0$$

Hinweis: Nutzen Sie die Monotonie der Ackermannfunktion in beiden Parametern aus.

Aufgabe T17

Sind WHILE-Programme immer noch Turing-mächtig, wenn die Zuweisungen $x_i := x_j + c$ nur noch für $c \in \{-1, 0\}$ erlaubt sind?

Aufgabe H16 (8 Punkte)

Betrachten Sie die nachfolgenden Varianten des Euklidischen Algorithmus (entnommen aus Wikipedia) zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen a und b.

Variante 1:

Eingabe: $a, b \in \mathbb{N}$
```
While a ≠ b
   If a > b
      then a := a − b
      else b := b − a
End While
```
Ausgabe: a

Variante 2:

Eingabe: $a, b \in \mathbb{N}$
```
While b > 0
    r := a mod b
    a := b
    b := r
End While
```
Ausgabe: a

1. Bestimmen Sie eine möglichst scharfe untere Schranke für die Worst-Case-Laufzeit von Variante 1 im uniformen Kostenmaß.
2. Bestimmen Sie eine möglichst scharfe obere Schranke für die Worst-Case-Laufzeit von Variante 2 im uniformen Kostenmaß.
3. Nutzen Sie Ihre Abschätzungen, um zu zeigen, daß sich die uniformen Worst-Case-Laufzeiten beider Varianten durch einen exponentiellen Faktor unterscheiden.
4. Stimmt diese Aussage auch bezüglich der Laufzeiten im logarithmischen Kostenmaß?

Aufgabe H17 (6 Punkte)
Geben Sie ein LOOP-Programm an, daß folgende Sprache akzeptiert:

$$L = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$$

wobei $|w|_i$ die Anzahl der Stellen in w angibt, an denen die Ziffer i steht.
Gehen Sie davon aus, daß die Eingabe in der Variable x_1 als Binärzahl kodiert steht und das zudem die Länge der Eingabe in der Variable x_2 zu finden ist. Wenn der Wert x_1 in der Sprache L enhalten ist, soll am Ende des Programs x_0 eine Eins enthalten, ansonsten Null.
Nehmen Sie an, daß die Subtraktion von Variablen mit dem Wert 0 wiederum 0 ergibt (Variablen können also nie negative Werte enthalten). Zudem setzen wir voraus, daß die Eingabe x_1 nie das leere Wort enthält.