Aufgabe 1 Berechenbare Funktionen

Welche der folgenden Funktionen ist berechenbar?

a) f: |N -> |R f(n) = n / 3
b) f: |N -> |R f(n) = n/3 mit n kodiert als Dezimalzahl
c) f: |N -> |R f(n) = $n^{½}$ für eine natürliche Zahlen n
d) f(n,k) = k-te Ziffer der Wurzel von n für eine natürliche Zahlen n

Begründen Sie jeweils ihre Antwort entweder durch Angabe eines Algorithmus oder Beweis für nicht Berechenbarkeit!

Lösung:

a) berechenbar für Darstellung des Ergebnisses z.B.
   - mit Periode: 8/3 = 2 Periode 3
   - als Ternärzahl: 22/ 10 = 2,1 (22 ist die Ternär-Darstellung von 8 = 2*3 +2)
   - mit Rest wie in Grundschule: 8/3 = 2 Rest 2
b) nicht berechenbar, da Dezimalzahl unendlich lang sein kein z.B. 8/3 = 2,666…
c) nicht berechenbar, da die reelle Zahl $2^{½}$ irrational ist und die Ziffernfolge sich nicht endlich beschreiben lässt.
d) Berechenbar. Die nächste Ziffer lässt sich durch Verfeinerung der Intervallgrenzen berechnen: Mit welcher Ziffer ist das Quadrat noch kleiner als 2 und mit Ziffer+1 größer als 2.

Verständnisaufgabe zum Diagonalisierungsbeweis

(i) Wenn Sie eine beliebige Dualzahl i nehmen und diese als Java-Programm interpretieren, dann ist die Wahrscheinlichkeit groß, dass diese Java-Programm nicht syntaktisch korrekt ist. Wie ist in diesem Fall die zugehörige Zeile im Diagonalisierungsbeweis definiert und wie reagiert die Diagonalisierungsfunktion?

(ii) Wenn Sie als Programmiersprache eine Teilmenge von Java verwenden und weder indefinite Schleifen noch Rekursion erlauben, welche Zeitkomplexität hätten diese Programme? Wäre in diesem Fall die Diagonalisierungsfunktion berechenbar?

Lösung

(i) In der Zeile eines syntaktisch nicht korrekten Programms steht überall undefiniert und das Javastop-Programm müsste 0 ausgeben (entsprechend auch die Diagonalfunktion an dieser Stelle gemäß Definition)

(ii) Diese Programme hätten konstanten Zeitaufwand
(bzw. bei definiten Schleifen jeweils O(n) bzw. $O(n^k)$ mit Schachtelungstiefe k)
und die Diagonalisierungsfunktion wäre berechenbar. Ein Simulator könnte diese Eigenschaft vorab prüfen.
Konsequenz: Diese Programmiersprache enthält **nicht alle intuitiv berechenbaren Funktionen** (wie z.B. die Diagonalisierungsfunktion). Insbesondere ist diese eingeschränkte Programmiersprache **nicht Turing-vollständig**!

Aufgabe 2 Entscheidbarkeit des Halteproblems

Für welche der folgenden Mengen ist das Halteproblem entscheidbar?

a) { (P,D) | Java-Program P enthält nur eine Folge von Anweisungen, d.h. ohne Schleifen, Rekursion, Goto-Sprünge }
b) { (P,D) | Java-Program P enthält keine Schleifen und bei Modulaufrufen keine Rekursion, sondern nur eine Hierarchie von Modulaufrufen}
c) { (P,D) | Java-Program P enthält nur Anweisungen, allerdings sind Sprünge erlaubt}

Lösung

a) Entscheidbar: Antwort ja (sogar mit konstantem Zeitaufwand), falls P diese Eigenschaften besitzt, und dies lässt sich mit Compilerbaumethoden überprüfen.
b) Entscheidbar: Antwort ja (sogar mit konstantem Zeitaufwand), falls P diese Eigenschaften besitzt, und dies lässt sich mit Compilerbaumethoden überprüfen.
c) Nicht entscheidbar, da jede Turingmaschine mit n Zuständen sich durch ein Java-Programm mit n if-Anweisungen mit GoTos simulieren lässt.