Sammlung aller Staatsexamensaufgaben der Prüfungsnummer

66110

Automatentheorie, Algorithmische Sprache (vertieft)

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Herbst 1989

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____ Kennwort: _____ Arbeitsplatz-Nr.: _____	HERBST 1989	66110

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)
Einzelprüfung: Automatentheorie, Algorithmische Sprachen
Anzahl der gestellten Themen (Aufgaben): 1
Anzahl der Druckseiten dieser Vorlage: 4

bitte wenden!

Sämtliche Teilaufgaben sind zu bearbeiten!

Teilaufgabe 1

Hinweis: Verwenden Sie zur Beschreibung der in dieser Aufgabe zu entwickelnden Algorithmen eine konkrete Programmiersprache wie PASCAL, MODULA o.ä. oder einen diesen Sprachen verwandten, in einschlägigen Vorlesungen oder Büchern üblicherweise benutzten "Pseudocode".

Gegeben sei der Zeichenvorrat $A = \{a, b, c\}$. M sei die Menge aller Zeichenreihen x über A mit der Eigenschaft, daß die Anzahl, wie oft das Zeichen **a** in x vorkommt, eine gerade Zahl ist.

a) Die Grammatik Γ habe A als Menge der Terminalzeichen, die Nichtterminalzeichen G und U, das Axiom G und die Produktionsregeln

$$G \longrightarrow \varepsilon$$
$$G \longrightarrow bG$$
$$G \longrightarrow cG$$
$$G \longrightarrow aU$$
$$U \longrightarrow bU$$
$$U \longrightarrow cU$$
$$U \longrightarrow aG$$

(ε bezeichne die leere Zeichenreihe.)

Beweisen Sie, daß für den Sprachschatz $\mathcal{L}(\Gamma)$ von Γ gilt: $\mathcal{L}(\Gamma) = M$.

b) Geben Sie einen endlichen Automaten an, der genau die Zeichenreihen von M akzeptiert.

c) Geben Sie eine kontextfreie Grammatik $\overline{\Gamma}$ an, für die ebenfalls $\mathcal{L}(\overline{\Gamma}) = M$ gilt, die jedoch weniger Produktionsregeln als Γ hat.

d) Formulieren Sie in Anlehnung an Γ einen rekursiven Algorithmus

function TESTM(string *s*):boolean;
...

der testet, ob eine Zeichenreihe s aus M ist oder nicht. Beweisen Sie, daß der Algorithmus für alle Eingaben s der Sorte string terminiert.

Hinweis: Nehmen Sie dabei an, daß die Datenstruktur string (der Zeichenreihen über einem Zeichenvorrat, der die Zeichen von A enthält) mit den Grundoperationen *isempty*, *first* und *rest* zur Verfügung steht. Dabei ist für eine Zeichenreihe $x = x_1x_2...x_n$:

$$isempty(x) = true \Leftrightarrow x = \varepsilon$$

und, falls $x \neq \varepsilon$:

$$first(x_1x_2...x_n) = x_1,$$
$$rest(x_1x_2...x_n) = x_2...x_n.$$

Fortsetzung nächste Seite!

Für $n \in \mathbb{N}_0$ sei nun M_n die Menge aller Zeichenreihen der Länge n aus M. Die Abbildung $h: \mathbb{N}_0 \to \mathbb{N}_0$ sei definiert als:

$$h(n) = \text{Anzahl der Elemente von } M_n.$$

e) Beweisen Sie, daß für h gilt:

$$h(n) = \begin{cases} 1, & \text{falls } n = 0, \\ 3^{n-1} + h(n-1) & \text{sonst.} \end{cases}$$

Hinweis: Es gibt genau 3^n Zeichenreihen der Länge n über A.

f) Formulieren Sie anhand von e) einen rekursiven Algorithmus zur Berechnung von $h(n)$ für gegebenes $n \in \mathbb{N}_0$. Nehmen Sie dabei an, daß Addition, Subtraktion und Multiplikation, nicht jedoch die Potenzierung als arithmetische Grundoperationen zur Verfügung stehen. Bestimmen Sie (in Abhängigkeit von n) die Anzahl der Additionen, Subtraktionen und Multiplikationen, die gemäß diesem Algorithmus bei der Berechnung von $h(n)$ durchgeführt werden.

g) Beweisen Sie, daß für $n > 0$ auch gilt:

$$h(n) = 3 * h(n-1) - 1.$$

Was bedeutet diese Beziehung im Hinblick auf die Komplexität der Berechnung von h gemäß f) ?

h) Für die Entrekursivierung der Berechnung von h wird durch die Beziehung aus g) eine Einbettung von h in die Abbildung $g: \mathbb{N}_0^3 \to \mathbb{Z}$ mit

$$g(n,k,m) = k * h(n) - m$$

nahegelegt.

Entwickeln Sie zunächst einen repetetiv rekursiven Algorithmus zur Berechnung von $g(n,k,m)$ für gegebene $n,k,m \in \mathbb{N}_0$ und daraus einen iterativen Algorithmus zur Berechnung von $h(n)$ für gegebenes $n \in \mathbb{N}_0$.

Teilaufgabe 2

Die Abbildungen *null*: $\mathbb{N}_0 \to \mathbb{N}_0$ und *succ*: $\mathbb{N}_0 \to \mathbb{N}_0$ seien definiert durch:

$$null(x) = 0,$$
$$succ(x) = x + 1.$$

Zu einer Abbildung $f: \mathbb{N}_0 \to \mathbb{N}_0$ seien $f^0: \mathbb{N}_0 \to \mathbb{N}_0$ und $f^1: \mathbb{N}_0 \to \mathbb{N}_0$ definiert durch:

$$f^0(0) = 0, \quad f^0(x+1) = f(f^0(x)),$$
$$f^1(0) = 1, \quad f^1(x+1) = f(f^1(x)).$$

Die Menge F von Abbildungen $\mathbb{N}_0 \to \mathbb{N}_0$ sei induktiv definiert wie folgt:

i) *null* und *succ* sind in F enthalten.
ii) Ist $f \in F$, so ist auch f^0 und f^1 in F enthalten.
iii) Sind $f,g \in F$, so ist auch h mit $h(x) = f(g(x))$ in F enthalten.

a) Bestimmen Sie $null^0$, $null^1$, $succ^0$ und $succ^1$ (in rekursionsfreier Darstellung).

Fortsetzung nächste Seite!

b) Beweisen Sie, daß die Abbildungen

$$\begin{aligned} &\mathit{eins}: \mathbb{N}_0 \to \mathbb{N}_0, && \mathit{eins}(x) = 1, \\ &\mathit{add2}: \mathbb{N}_0 \to \mathbb{N}_0, && \mathit{add2}(x) = 2+x, \\ &\mathit{mult2}: \mathbb{N}_0 \to \mathbb{N}_0, && \mathit{mult2}(x) = 2*x, \\ &\mathit{pot2}: \mathbb{N}_0 \to \mathbb{N}_0, && \mathit{pot2}(x) = 2^x \end{aligned}$$

in F enthalten sind.

c) Beweisen Sie: Ist $f \in F$, so ist auch die Abbildung $r: \mathbb{N}_0 \to \mathbb{N}_0$ mit

$$r(x) = \begin{cases} 1, & \text{falls } 2^{f(x)}+2 > 3, \\ 0 & \text{sonst} \end{cases}$$

in F enthalten.

Zu einer Abbildung $h: \mathbb{N}_0 \to \mathbb{N}_0$ sei nun die Abbildung $ack_h: \mathbb{N}_0 \to \mathbb{N}_0$ ("verallgemeinerte Ackermann-Funktion") definiert durch:

$$\begin{aligned} &ack_h(0,y) = h(y), \\ &ack_h(x+1,0) = ack_h(x,1), \\ &ack_h(x+1,y+1) = ack_h(x, ack_h(x+1,y)). \end{aligned}$$

Die unendliche Folge $h_0, h_1, h_2, \ldots$ von Abbildungen $\mathbb{N}_0 \to \mathbb{N}_0$ sei gegeben durch:

$$h_i(y) = ack_h(i,y) \qquad \text{für } i = 0,1,2,\ldots$$

d) Beweisen Sie: Falls $h(0) \neq 0$ und $h(y+1) > h(y)$ für alle $y \in \mathbb{N}_0$, so gilt für alle $i \in \mathbb{N}_0$ und $y \in \mathbb{N}_0$:

d1) $h_i(y) > y$,
d2) $h_i(y+1) > h_i(y)$,
d3) $h_{i+1}(y) \geq h_i(y+1)$,
d4) $h_{i+1}(y) > h_i(y)$.

e) Beweisen Sie: Falls $h \in F$, so gilt $h_i \in F$ für alle $i \in \mathbb{N}_0$.

f) Sei $h \in F$. Sind die Abbildungen h_i ($i \in \mathbb{N}_0$) dann primitiv-rekursiv ? Begründen Sie Ihre Antwort.

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Frühjahr 1990

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____		
Kennwort: _____	FRÜHJAHR 1990	66110
Arbeitsplatz-Nr.: _____		

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie, Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 3

bitte wenden!

Sämtliche Teilaufgaben sind zu bearbeiten!

Teilaufgabe 1

Gegeben sei ein endliches Alphabet A und eine ungeordnete, endliche, nichtzyklische Liste von K Paaren (n, t) für ein vorgegebenes $K \in \mathbb{N}$, worin die n nichtleere endliche Zeichenreihen aus $A^* \setminus \{\epsilon\}$ und die t natürliche Zahlen aus $\mathbb{N}$ seien. In A^* steht die lexikographische Ordnung zur Verfügung, die zur Unterscheidung von der Ordnung $<$ in $\mathbb{N}$ mit $\sqsubset$ bezeichnet werde. Außerdem gelte für alle n in den Paaren der Liste: $|n| \leq L$ für ein vorgegebenes $L \in \mathbb{N}$, wobei mit $|x|$ die Länge einer Zeichenreihe $x \in A^*$ bezeichnet wird.
Die Paare der Liste können als einfache Karteikarten in einer Telefondatei aufgefaßt werden mit der Bedeutung:

$$n \triangleq \text{Name}$$
$$t \triangleq \text{Telefonnummer.}$$

Es wird vorausgesetzt, daß für verschiedene Paare (n_i, t_i) und (n_j, t_j) in der Liste gilt: $n_i \neq n_j$ und $t_i \neq t_j$.

1. Geben Sie Datenstrukturen durch Typ- und Identitätsvereinbarungen an, mit denen die folgenden Teilaufgaben bearbeitet werden können.

2. Formulieren Sie einen Algorithmus, mit dessen Hilfe eine Zugriffsstruktur auf die Liste aufgebaut wird. Die Zugriffsstruktur soll es ermöglichen, zu einem Namen n mit der Komplexität $O(\log K)$

   (a) zu entscheiden, ob die Liste einen Eintrag zu n enthält, und

   (b) gegebenenfalls die zugehörige Telefonnummer t anzugeben.

3. Schreiben Sie eine Prozedur *zugriff* in PASCAL, die den unter 2. formulierten Algorithmus realisiert.

4. Schreiben Sie eine Prozedur *suche*, die mit Hilfe der unter 3. aufgebauten Zugriffsstruktur zu einem $n \in A^*$ feststellt, ob die Liste einen Eintrag zu n enthält, und gegebenenfalls die zugehörige Telefonnummer t ausgibt. Die Komplexität der Prozedur *suche* soll $O(\log K)$ sein.

Fortsetzung nächste Seite!

Teilaufgabe 2

Gegeben sei das Alphabet $A = \{a, b\}$. Mit x_a bzw. x_b werde für ein $x \in A^*$ die Zeichenreihe aus $\{a\}^*$ bzw. $\{b\}^*$ bezeichnet, die durch Streichen aller b bzw. a aus x entsteht. Seien also z.B. $x = aabab$ und $y = bbbb$, dann ist $x_a = aaa$, $x_b = bb$, $y_a = \epsilon$ und $y_b = bbbb$. Gegeben sei nun die wie folgt definierte Teilmenge M von A^*:

$$x \in M \quad \Leftrightarrow_{df} \quad |x_a| \leq |x_b|,$$

wobei für ein $x \in A^*$ mit $|x|$ die Länge von x bezeichnet wird.

1. Zeigen Sie, daß die Menge $M \subset A^*$ nicht regulär ist.
2. M ist als Sprachschatz einer kontextfreien Sprache über dem terminalen Alphabet A darstellbar. Beweisen Sie diese Aussage dadurch, daß Sie einen Kellerautomaten angeben, von dem Sie zeigen, daß er genau die Menge M akzeptiert.
3. Konstruieren Sie eine kontextfreie Grammatik über dem terminalen Alphabet A, die in A^* genau die Menge M erzeugt, und begründen Sie die einzelnen Schritte Ihres konstruktiven Vorgehens.

Teilaufgabe 3

Gegeben seien zwei ganze Zahlen p und q mit $0 < q < p$ und eine wie folgt definierte rekursive Rechenvorschrift f für ganze Zahlen $z \in \mathbb{Z}$:

$$f(z) := \begin{cases} f(f(z-p)) & \text{für } z \geq 100 \\ z+q & \text{für } z < 100 \end{cases}$$

Beweisen Sie, daß die Rechenvorschrift f für alle $z \in \mathbb{Z}$ terminiert und somit eine Funktion

$$f : \mathbb{Z} \to \mathbb{Z}$$

definiert.

Hinweis:
Betrachten Sie für $z \geq 100$ die durch $f(z)$ veranlaßten rekursiven Aufrufe $f(z_i)$ von f und zeigen Sie, daß für alle i gilt: $z_i < z$.

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Herbst 1990

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	HERBST	66110
Kennwort: _____	1990	
Arbeitsplatz-Nr.: _____		

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie, Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 3

Sämtliche Teilaufgaben sind zu bearbeiten!

Teilaufgabe 1

Gegeben sei das Alphabet $A=\{a,b,c\}$. Die Mengen M_a, M_b, M_c und M von Zeichenreihen über A seien definiert durch

$$M_x = \{w \in A^* \mid w = uxxv \text{ mit } u,v \in A^*\} \quad \text{für } x = a, b \text{ bzw. } c,$$
$$M = M_a \cup M_b \cup M_c.$$

a) Beschreiben Sie M durch einen regulären Ausdruck!

b) Geben Sie einen deterministischen endlichen Automaten an, der genau die Zeichenreihen von M akzeptiert!

c) Geben Sie eine reguläre Grammatik an, die M als Sprachschatz hat!

Fortsetzung nächste Seite

Teilaufgabe 2

Gegeben sei die Grammatik Γ mit {a,b} als Menge der Terminalzeichen, den Nichtterminalzeichen **Z,A,B**, dem Axiom **Z** und den Produktionsregeln

Z → AB
A → ZA
A → a
B → ZB
B → b

a) Zeigen Sie, daß die Zeichenreihe **aabbabab** zum Sprachschatz von Γ gehört!

b) Überführen Sie Γ in die Greibach-Normalform!

c) Geben Sie einen (gegebenenfalls nicht-deterministischen) Kellerautomaten an, der genau den Sprachschatz von Γ akzeptiert!

Teilaufgabe 3

Durch die Funktionsvereinbarung

function $h(m,n$:**nat**)**nat**:
  if $m=0$ **then** n **else** $2 \cdot h(m-1,n)$ **endif**

ist eine Funktion $h:\mathbb{N}_0^2 \to \mathbb{N}_0$ definiert.

Beweisen Sie

a) durch Berechnungsinduktion,

b) durch Parameterinduktion (nach m),

daß für alle $m,n \in \mathbb{N}_0$ gilt:

$$h(m,2 \cdot n) = 2 \cdot h(m,n).$$

Fortsetzung nächste Seite!

Teilaufgabe 4

Durch die Funktionsvereinbarung

```
function f(n:nat)nat:
  if n≤2 then n else 2•f(n-1)+f(n-3) endif
```

ist eine Funktion $f:\mathbb{N}_0 \to \mathbb{N}_0$ definiert.

a) Beweisen Sie, daß f für alle $n \in \mathbb{N}_0$ terminiert!

b) Die Funktion $g:\mathbb{N}_0^4 \to \mathbb{N}_0$ sei gegeben durch

$$g(n,x,y,z) = x \cdot f(n+2) + y \cdot f(n+1) + z \cdot f(n).$$

Beweisen Sie:

$$g(n,x,y,z) = \begin{cases} 2 \cdot x + y, & \text{falls } n=0 \\ g(n-1, 2 \cdot x + y, z, x), & \text{falls } n>0. \end{cases}$$

c) Entwickeln Sie mit Hilfe von b) zunächst einen repetetiv rekursiven Algorithmus zur Berechnung von $g(n,x,y,z)$ für gegebene $n,x,y,z \in \mathbb{N}_0$ und daraus einen iterativen Algorithmus zur Berechnung von $f(n)$ für gegebenes $n \in \mathbb{N}_0$!

Hinweis: Formulieren Sie die Algorithmen in einer Programmiersprache wie PASCAL, MODULA o.ä. oder in einem "Pseudocode", wie er in obiger Funktionsvereinbarung verwendet ist!

Teilaufgabe 5

Für $r \in \mathbb{R}$ bezeichne $[r]$ die (eindeutig bestimmte) ganze Zahl z mit $z \le r < z+1$. Die Funktion $f:\mathbb{N}_0 \to \mathbb{N}_0$ sei gegeben durch

$$f(x) = [3 \cdot \sqrt{x}].$$

Beweisen Sie, daß f primitiv-rekursiv ist!

Hinweis: Die üblichen arithmetischen und Booleschen Operationen wie $+,\cdot,\le,<,\wedge,\vee$ u.ä. dürfen als primitiv-rekursiv vorausgesetzt werden.

Teilaufgabe 6

A und B seien zwei rekursiv aufzählbare Teilmengen von $\mathbb{N}_0$ mit $A \cup B = \mathbb{N}_0$.

Beweisen Sie: Falls $A \cap B$ rekursiv ist, so sind A und B rekursiv!

Hinweis: Für $M,N \subseteq \mathbb{N}_0$ gilt:
a) M ist genau dann rekursiv, wenn M und $\mathbb{N}_0 \setminus M$ rekursiv aufzählbar ist.
b) Falls M und N rekursiv aufzählbar sind, so ist $M \cap N$ rekursiv aufzählbar.

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Herbst 1991

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	HERBST	66110
Kennwort: _____	1991	
Arbeitsplatz-Nr.: _____		

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie, Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 4

bitte wenden!

Die gesamte Prüfungsaufgabe besteht aus den nachfolgenden Aufgaben 1 - 4.

Aufgabe 1

Gegeben seien das Alphabet $A = \{a,b\}$ sowie folgende Mengen M_1 und M_2:

M_1 = Menge aller Zeichenreihen über A, die mindestens ein Paar aufeinanderfolgender Zeichen **a** enthalten,

M_2 = Menge aller Zeichenreihen über A, die höchstens ein Paar aufeinanderfolgender gleicher Zeichen enthalten.

a) Geben Sie eine reguläre Grammatik an, die M_1 als Sprachschatz hat!

b) Geben Sie einen deterministischen endlichen Automaten an, der genau die Zeichenreihen von M_2 akzeptiert!

c) Geben Sie einen regulären Ausdruck an, der eine Menge L von Zeichenreihen über A beschreibt, für die gilt:

$$M_1 = LA^*.$$

Beweisen Sie Ihre Behauptung!

d) Beweisen oder widerlegen Sie: Es gibt eine Menge N von Zeichenreihen über A mit

$$M_2 = NA^*.$$

Aufgabe 2

M sei die Menge aller Zeichenreihen w über dem Alphabet $\{0,1\}$ mit der Eigenschaft, daß w doppelt so viele Zeichen 1 wie Zeichen 0 enthält.

Geben Sie eine Turingmaschine an, die genau die Menge M akzeptiert!

Aufgabe 3

Hinweis: Verwenden Sie zur Beschreibung der in dieser Aufgabe zu entwickelnden Algorithmen eine Syntax, wie sie in höheren Programmiersprachen wie PASCAL, MODULA o.ä. üblich ist.

Für die Menge bbchar aller Binärbäume über einer Grundmenge char von Zeichen seien als Grundoperationen verfügbar:

empty: → bbchar,	*empty* = leerer Binärbaum,
isempty: bbchar → boolean,	*isempty*(b) = *true* ⇔ b ist leer,
root: bbchar → char,	*root*(b) = Wurzel von b, falls $b \neq$ *empty*,
left: bbchar → bbchar,	*left*(b) = linker Unterbaum von b, falls $b \neq$ *empty*,
right: bbchar → bbchar,	*right*(b) = rechter Unterbaum von b, falls $b \neq$ *empty*.

(Für b = *empty* ist *root*(b), *left*(b), *right*(b) jeweils undefiniert.)

Fortsetzung nächste Seite!

a) Definieren Sie mit Hilfe dieser Grundoperationen rekursiv die folgenden weiteren Operationen (wobei diese Definitionen gegebenenfalls auf weitere geeignet definierte Operationen abgestützt werden können):

a1) *bbgleich*: bbchar × bbchar → boolean,
bbgleich(b_1,b_2) = *true* ⇔ b_1 und b_2 sind gleich,

a2) *istord*: bbchar → boolean, *istord*(b) = *true* ⇔ b ist geordnet (sortiert),

a3) *istvoll*: bbchar → boolean, *istvoll*(b) = *true* ⇔ b ist vollständig.

b) Die Operation *enthalten*: bbchar × char → boolean mit

enthalten(b,x) = *true* ⇔ x ist als Knoten in b enthalten

kann rekursiv wie folgt definiert werden:

enthalten(b,x) = **If** *isempty*(b) **then** *false*
else x = *root*(b) ∨ *enthalten*(*left*(b),x) ∨ *enthalten*(*right*(b,x))
endif

Geben Sie - unter Verwendung einer geeignet gewählten Datenstruktur keller (für Kellerspeicher) - einen iterativen Algorithmus an, der *enthalten*(b,x) für gegebene b und x berechnet!

c) Unter der Voraussetzung, daß b geordnet (sortiert) ist, läßt sich *enthalten* linear rekursiv definieren.

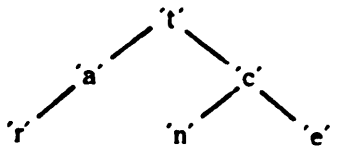
Geben Sie diese Definition und einen entsprechenden iterativen Algorithmus (ohne Verwendung eines Kellers) zur Berechnung an!

d) Binärbäume seien nun in üblicher Weise durch Geflechte realisiert, in PASCAL-Notation etwa:

```
TYPE bbchar = ↑bbelem;
     bbelem = RECORD
                wurzel : char;
                lub : bbchar;     (* linker Unterbaum *)
                rub : bbchar      (* rechter Unterbaum *)
              END;
```

Geben Sie Algorithmen zur Realisierung der Operationen *isempty*, *root* und *left* gemäß dieser Darstellung an!

e) Geben Sie einen Algorithmus an, der den Binärbaum



in der Darstellung von Teilaufgabe d) erzeugt!

Geben Sie dazu zunächst einen Algorithmus für die Operation

compose: char × bbchar × bbchar → bbchar,
compose(x,b_1,b_2) = Binärbaum mit Wurzel x, linkem Unterbaum b_1 und rechtem Unterbaum b_2

an und verwenden Sie diesen zum Aufbau des Binärbaums!

Fortsetzung nächste Seite!

Aufgabe 4

Durch die Funktionsvereinbarung

```
function f(x,y,z:nat)nat:
  if x = y then z else f(x,y+1,(y+1)*z) endif
```

ist eine Funktion $f: \mathbb{N}_0^3 \to \mathbb{N}_0$ gegeben.

a) Bestimmen Sie den Wert von $f(4,0,2)$!

b) Beweisen Sie: $f(x,y,z)$ terminiert für alle $x,y,z \in \mathbb{N}_0$ mit $x \geq y$!

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Frühjahr 1993

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____ Kennwort: _____ Arbeitsplatz-Nr.: _____	**FRÜHJAHR 1993**	**66110**

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie, Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 4

bitte wenden!

Sämtliche Teilaufgaben sind zu bearbeiten!

1. Formale Sprachen

(1a) Sei $G = (T, N, P, S)$ eine Chomsky-Grammatik. Was bedeuten die angegebenen Komponenten? Wie sind die Produktionenmenge P und der Ableitbarkeitsbegriff definiert?

(1b) Welche formalen Einschränkungen auf P führen zu den monotonen, kontextsensitiven und kontextfreien Grammatiken? Wie liegen die Klassen der erzeugten Wortmengen zueinander? Soweit sie nicht gleich sind, geben Sie (ohne Beweis) je ein typisches Beispiel aus der Differenzmenge an!

(1c) Für kontextfreie Grammatiken gilt das Zerlegungslemma. Formulieren und beweisen Sie dieses Lemma!

(1d) Beweisen Sie, daß sich die Wortmenge

$$\{a^i b^j \mid i, j \geq 0 \wedge i \neq j\}$$

von einer kontexfreien Grammatik erzeugen läßt!

2. Berechenbarkeit

(2a) Der Berechenbarkeitsbegriff kann nach Kleene durch rekursive Funktionen definiert werden. Geben Sie die Definition der primitiv-rekursiven und der μ-rekursiven Funktionen an!

(2b) Zeigen Sie, daß der beschränkte μ-Operator nicht aus dem Bereich der primitiv-rekursiven Funktionen hinausführt!

(2c) Beweisen Sie, daß die ganzzahlige Quadratwurzelfunktion $\lfloor\sqrt{n}\rfloor$ primitiv-rekursiv ist!

(2d) Die μ-rekursiven Funktionen lassen sich unmittelbar in *while*-Programme übersetzen, die nur aus

- Wertzuweisungen,
- dem Aufruf der Nachfolger- und Vorgängerfunktion,
- while-Schleifen und
- Prozeduraufrufen

bestehen. Zeigen Sie, daß es kein while-Programm mit genau einer Variablen gibt, das die Funktion $f(x) = 2x$ berechnet!

Fortsetzung nächste Seite!

3. Algorithmische Sprachen

(3a) Eine Objektart ist durch eine Trägermenge und einen Satz Operationen definiert. Wie entsteht die Trägermenge bei einer Verbundart und welche Operationen sind darauf definiert? Verwenden Sie als Beispiel:

```
mode datum = (int [1..31] tag,
              int [1..12] monat,
              int [1900..1999] jahr)
```

(3b) Definieren Sie die Objektart der binären Bäume als rekursive Verbundart! Mit welchem Konzept realisiert man rekursive Verbundarten in den gängigen Programmiersprachen (PASCAL, C)?

(3c) Beschreiben Sie eine Technik zur Umwandlung beliebiger Bäume in binäre! Zeigen Sie durch eine Plausibilitätsbetrachtung, daß das Durchlaufen des Baumes in Prä-Ordnung von dieser Umwandlung nicht berührt wird!

(3d) Welche Konstruktionsvorschrift liegt den Artvarianten zugrunde? Erläutern Sie Ihre Antwort an einem geeigneten Beispiel! Zeigen Sie, daß die unmittelbare Verwendung des zugrundeliegenden mathematischen Konzeptes eine statische Typprüfung nicht zuläßt!

4. Programmiermethodik

Betrachten Sie folgende Spezifikation eines abstrakten Datentyps:

```
init:                    -->  type
add:        type x nat   -->  type
remove:     type         -->  type
is_empty:   type         -->  boolean
first:      type         -->  nat
is_empty(init)             =  true
is_empty(add(t,n))         =  false
first(add(init,n))         =  n
remove(add(init,n))        =  init
first(add(add(t,n),m))     =  first(add(t, n))
remove(add(add(t,n),m))    =  add(remove(add(t,n)),m)
```

(4a) Erläutern Sie unter Verwendung dieses Beispiels, was die *wesentliche* Idee der algebraischen Spezifikation ist!

(4b) Die angegebene Spezifikation definiert einen in der Informatik häufig anzutreffenden Datentyp. Welcher Datentyp ist das? Begründung!

Fortsetzung n.. ste Seite!

(4c) Das angegebene Gleichungssystem induziert auf der Termalgebra eine Kongruenzrelation. Geben Sie deren formale Definition an!

(4d) Zeigen Sie, daß es in jeder der durch (c) definierten Kongruenzklassen genau einen Term gibt, der entweder die Operation *remove* überhaupt nicht oder nur in der Form *remove(init)* enthält!

5. Übersetzerbau

(5a) Was versteht man unter einem Compilergenerator? Erläutern Sie Aufgabe, Aufbau und Datenfluß!

(5b) Als Eingabe für einen Compilergenerator reicht eine kontextfreie Grammatik nicht aus. Wie werden die nichtkontextfreien Aspekte der Syntax und die Semantik beim Compilergenerator *Yacc* berücksichtigt?

(5c) Betrachten Sie folgende Grammatik für Schleifen:

```
WHILE_stat    ::=  WHILE b LOOP seq_of_stat END;
seq_of_stat   ::=  statement  | seq_of_stat statement
statement     ::=  WHILE_stat | other_stat;
```

Verändern und/oder ergänzen Sie die erste dieser Produktionen, so daß *Yacc* die Sprunganweisungen, die zur Realisierung einer Schleife nötig sind, an der richtigen Stelle erzeugt. Begründen Sie Ihre Änderung!

(5d) Bei Erreichen eines Syntaxfehlers kommt es darauf an, die Syntaxanalyse so fortzusetzen, daß möglichst wenige Folgefehler auftreten. Beschreiben Sie das auf Graham und Rhodes zurückgehende Verfahren zum Wiederaufsetzen! Ist das Verfahren im *Yacc* einsetzbar? (Hinweis: Bei der Beschreibung spielen die folgenden Begriffe eine Rolle: Kondensierung,. Rückwärts-, Vorwärtsschritte.)

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Herbst 1993

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**HERBST**	**66110**
Kennwort: _____	**1993**	
Arbeitsplatz-Nr.: _____		

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie, Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 3

bitte wenden!

Sämtliche Teilaufgaben sind zu bearbeiten!

Hinweis: Verwenden Sie zur Formulierung von Algorithmen eine Programmiersprache wie PASCAL, MODULA o.ä. oder einen "Pseudocode", wie er in einschlägigen Vorlesungen und Büchern üblicherweise benutzt wird!

Aufgabe 1

Gegeben sei die Grammatik Γ mit $\Sigma = \{a,b\}$ als Menge der Terminalzeichen, den Nichtterminalzeichen Z, A und B, dem Axiom Z und den Produktionsregeln

$Z \rightarrow a$
$Z \rightarrow aB$
$Z \rightarrow Aa$
$A \rightarrow ab$
$A \rightarrow aBb$
$A \rightarrow abA$
$B \rightarrow ba$

a) Beweisen Sie: Γ ist mehrdeutig.

b) Beweisen Sie: Für den Sprachschatz $\mathcal{L}(\Gamma)$ von Γ gilt:

$$\mathcal{L}(\Gamma) = \{a(ba)^n : n \in \mathbb{N}_0\}.$$

c) Geben Sie eine reguläre Grammatik an, die den gleichen Sprachschatz hat wie Γ.

d) Geben Sie einen deterministischen endlichen Automaten an, der genau die Zeichenreihen von $\mathcal{L}(\Gamma)$ akzeptiert!

Aufgabe 2

Gegeben sei das Alphabet $\Sigma = \{a,b\}$. Für eine Zeichenreihe $w \in \Sigma^*$ bezeichne $A(w)$ die Anzahl der Zeichen a in w und $B(w)$ die Anzahl der Zeichen b in w.

Die Menge $M \subseteq \Sigma^*$ sei definiert durch

$$M = \{w \in \Sigma^* : A(w) \text{ ist gerade, und } B(w) \text{ ist ungerade}\}.$$

Beweisen Sie: M ist entscheidbar. Geben Sie dazu eine Turing-Maschine an, die für alle $x \in \Sigma^*$ anhält und genau alle $x \in M$ akzeptiert!

Fortsetzung nächste Seite!

Aufgabe 3

Gegeben sei eine Funktion $G: \mathbb{N}_0 \to \mathbb{N}_0$ mit $G(x) \leq x$ für alle $x \in \mathbb{N}_0$.

Durch die Funktionsvereinbarung

```
function F(x:nat)nat:
  if x mod 2 = 0 then G(x) else F(F(x-1)) endif
```

ist eine Funktion $F: \mathbb{N}_0 \to \mathbb{N}_0$ definiert.

a) Beweisen Sie:

a1) $F(x)$ terminiert für alle $x \in \mathbb{N}_0$.

a2) Falls $G(x) = x$ für alle $x \in \mathbb{N}_0$, so gilt $F(F(x)) = F(x)$ für alle $x \in \mathbb{N}_0$.

b) Die Funktion $F^*: \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0$ sei wie folgt definiert:

$$F^*(x,y) = \begin{cases} x, & \text{falls } y = 0, \\ F^*(F(x), y-1) & \text{sonst.} \end{cases}$$

b1) Beweisen Sie: Für alle $x \in \mathbb{N}_0$ gilt $F(x) = F^*(x,1)$.

b2) Geben Sie eine repetitiv rekursive Funktionsvereinbarung für F^* an (die sich nicht auf F abstützt), und entwickeln Sie daraus durch Entrekursivierung und Spezialisierung (gemäß Teilaufgabe b1) einen iterativen Algorithmus zur Berechnung von F.

Aufgabe 4

Gegeben seien folgende Produktionsregeln (in Backus-Naur-Form) für die Syntaxdefinition von *Gleitpunktzahlen* (über dem Alphabet {+,-,.,E,0,1,2,3,4,5,6,7,8,9}):

```
<Gleitpunktzahl> → <Vorzeichen><nichtnegative Zahl> | <nichtnegative Zahl>
<nichtnegative Zahl> → <Mantisse><Exponent>
<Mantisse> → <Ziffernfolge>.<Ziffer><Ziffernfolge>
<Exponent> → E<ganze Zahl> | ε
<ganze Zahl> → <Vorzeichen><Ziffer><Ziffernfolge> | <Ziffer><Ziffernfolge>
<Ziffernfolge> → <Ziffer><Ziffernfolge> | ε
<Vorzeichen> → + | -
<Ziffer> → 0|1|2|3|4|5|6|7|8|9
```

a) Geben Sie für die Gleitpunktzahl

4.538E-2

den Syntaxbaum gemäß dieser Definition an!

b) Geben Sie einen Syntaxanalyse-Algorithmus nach der Methode des rekursiven Abstiegs ("recursive descent") an, der feststellt, ob eine vorgelegte Zeichenreihe eine Gleitpunktzahl gemäß dieser Definition ist!

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Frühjahr 1994

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____		
Kennwort: _____	FRÜHJAHR 1994	66110
Arbeitsplatz-Nr.: _____		

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie, Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 3

bitte wenden!

Sämtliche Teilaufgaben sind zu bearbeiten!

1. Formale Sprachen/Automatentheorie

(1a) Geben Sie eine kontextfreie Grammatik zu der Wortmenge

$$L = \{a^i b^j \mid i \geq j \geq 0\}$$

an.

(1b) Eine kontextfreie Grammatik G heißt reduziert, wenn jedes nichtterminale Symbol von G in einer aus dem Startsymbol ableitbaren Kette vorkommt und aus jedem nichtterminalen Symbol eine terminale Kette ableitbar ist. Geben Sie ein Verfahren an, das zu jeder kontextfreien Grammatik eine äquivalente reduzierte Grammatik liefert.

(1c) Beweisen Sie, daß die nichtdeterministischen, erkennenden endlichen Automaten nicht mehr können als die deterministischen.

(1d) Konstruieren Sie einen deterministischen, erkennenden endlichen Automaten, der die Wortmenge über dem Alphabet $\{a, b\}$ akzeptiert, die aus allen Wörtern besteht, die jede der folgenden Bedingungen erfüllen:

i. Die Länge des Wortes ist durch 3 teilbar.

ii. Das Wort beginnt mit a und endet mit b.

iii. aaa ist kein Teilwort des Wortes.

2. Berechenbarkeit/Algorithmische Sprachen

(2a) While-Programme bestehen aus Anweisungen der Form:

```
X := 0;
X := succ(Y);
X := pred(Y);
while X /= Y do Folge_von_Anweisungen end;
```

wobei X und Y beliebige Variablen, *succ* die Nachfolgerfunktion und *pred* die Vorgängerfunktion bezeichnet. Schreiben Sie ein While-Programm, das die Addition von n und m realisiert.

(2b) Was versteht man unter einer primitiv-rekursiven Funktion? Zeigen Sie, daß man jede primitiv-rekursive Funktion durch ein While-Programm realisieren kann, das stets hält.

Fortsetzung nächste Seite!

(2c) Wir nennen eine reelle Zahl α *näherungsweise berechenbar*, wenn es eine berechenbare Funktion $f(n)$ gibt, die für jeden Parameter n die ersten n Stellen der Dezimalentwicklung von α liefert. Zeigen Sie, daß die Quadratwurzel jeder natürlichen Zahl näherungsweise berechenbar ist. (Die Berechenbarkeit der arithmetischen Grundoperationen sei bekannt.)

(2d) Zeigen Sie, daß es reelle Zahlen gibt, die *nicht* näherungsweise berechenbar sind.

3. Programmiermethodik

(3a) Erläutern Sie die auf Floyd und Hoare zurückgehende Verifikationsmethode. (In Ihrer Antwort müssen mindestens die Begriffe Zusicherung, schwächste Vorbedingung und Prädikattransformation vorkommen.)

(3b) Betrachten Sie folgendes Programmfragment mit der Nachbedingung $a = n^3$:

```
a := 0;  b := 1;  c := 0;
WHILE c < 6*n DO
   a := a + b;
   c := c + 6;
   b := b + c;
END;
```

Geben Sie die Schleifeninvariante an, und beweisen Sie diese durch Induktion. Beweisen Sie als zweites die Nachbedingung. Was fehlt dann noch für eine vollständige Verifikation?

4. Übersetzerbau

(4a) Welches sind die drei wichtigsten Teilaufgaben, die ein Codegenerator *in jedem Fall* zu lösen hat? (Hinweis: Die Optimierung gehört nicht dazu.) Beschreiben Sie diese Aufgaben.

(4b) Generieren Sie Maschinencode zu den beiden folgenden C-Anweisungen:

```
x = a/(b+c) - d*(e+f);
a[i][j] = b[i][k] * c[k][j];
```

Gehen Sie dabei von einer Zielmaschine mit drei universell verwendbaren Registern aus.

(4c) Erläutern Sie die folgenden Parameterübergabemechanismen: Call-by-value, Call-by-reference, Call-by-name und Copy-Restore.

(4d) Beim Aufruf einer Prozedur muß eine Umgebungsbeschreibung angelegt werden (activation record). Welche Information muß diese Beschreibung bei pascalähnlichen Sprachen enthalten? Inwiefern vereinfacht sie sich bei Sprachen, die keine geschachtelten Prozedurdeklarationen zulassen?

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Herbst 1994

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 1994

66110

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie,Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 4

Bitte wenden!

Sämtliche Aufgaben sind zu bearbeiten!

Aufgabe 1

M sei die Menge aller Zeichenreihen über dem Alphabet {0,1}, die mindestens so viele Zeichen 1 wie Zeichen 0 enthalten.

Geben Sie eine (deterministische) Turingmaschine T an, die außer einem Leerzeichen nur die Zeichen aus {0,1} verwendet und M in folgendem Sinne akzeptiert: Ein Wort $w \in \{0,1\}^*$ steht auf dem ansonsten leeren Band! Angesetzt auf das erste Zeichen von w (bzw. auf ein Leerzeichen, falls w das leere Wort ist), erreicht T genau dann nach endlich vielen Schritten einen Endzustand, wenn $w \in M$ ist.

Aufgabe 2

Gegeben sei die Grammatik Γ_1 mit {a,b} als Menge der Terminalzeichen, den Nichtterminalzeichen Z,A,B, dem Axiom Z und den Produktionsregeln

$$\begin{aligned} Z &\to aA \\ A &\to b \\ A &\to bB \\ B &\to a \\ B &\to b \\ B &\to bB \end{aligned}$$

Die Grammatik Γ_2 entstehe aus Γ_1 dadurch, daß zu diesen Produktionsregeln noch die weitere Regel

$$B \to BA$$

hinzugenommen wird.

Der jeweilige Sprachschatz von Γ_1 und Γ_2 sei mit $\mathcal{L}(\Gamma_1)$ bzw. $\mathcal{L}(\Gamma_2)$ bezeichnet.

a) Beweisen Sie: $\mathcal{L}(\Gamma_1) = \{ab^n \mid n \in \mathbb{N}\} \cup \{ab^n a \mid n \in \mathbb{N}\}$.

b) Geben Sie einen deterministischen endlichen Automaten an, der genau die Zeichenreihen von $\mathcal{L}(\Gamma_1)$ akzeptiert.

c) Beweisen Sie: Γ_2 ist mehrdeutig.

d) Beweisen Sie: $\mathcal{L}(\Gamma_1) \neq \mathcal{L}(\Gamma_2)$.

e) Überführen Sie Γ_2 in Greibach-Normalform.

Aufgabe 3

Sei int die Menge der ganzen Zahlen und **sequ** int die Menge aller endlichen Folgen ganzer Zahlen. Für **sequ** int seien als Grundoperationen verfügbar:

Fortsetzung nächste Seite!

empty: → **sequ** int,	*empty* = leere Folge
isempty: **sequ** int → boolean,	*isempty*(*s*) = *true* ⇔ *s* ist leer
first: **sequ** int → int,	*first*: $(s_1,\ldots,s_n) \mapsto s_1$
rest: **sequ** int → **sequ** int,	*rest*: $(s_1,s_2\ldots,s_n) \mapsto (s_2,\ldots,s_n)$
prefix: int × **sequ** int → **sequ** int,	*prefix*: $(x,(s_1,\ldots,s_n)) \mapsto (x,s_1,\ldots,s_n)$

(Für *s* = *empty* sind *first*(*s*) und *rest*(*s*) nicht definiert.)

S_3 sei die Menge aller Folgen aus **sequ** int mit einer durch 3 teilbaren Anzahl von Komponenten.

a) Geben Sie (unter ausschließlicher Verwendung der genannten Grundoperationen) rekursive Funktionsvereinbarungen an für Funktionen *last*, *lead*, *postfix*, *conc* mit folgender Bedeutung (*last*(*s*) und *lead*(*s*) sind nur für *s* ≠ *empty* definiert):

last: **sequ** int → int,	*last*: $(s_1,\ldots,s_n) \mapsto s_n$
lead: **sequ** int → **sequ** int,	*lead*: $(s_1,\ldots s_{n-1},s_n) \mapsto (s_1,\ldots,s_{n-1})$
postfix: **sequ** int × int → **sequ** int,	*postfix*: $((s_1,\ldots,s_n),x) \mapsto (s_1,\ldots,s_n,x)$
conc: **sequ** int × **sequ** int → **sequ** int,	*conc*: $((s_1,\ldots,s_n),(t_1,\ldots,t_m)) \mapsto (s_1,\ldots,s_n,t_1,\ldots,t_m)$

b) Gegeben sei die Funktion *vorn* durch die Funktionsvereinbarung

```
function vorn(s:sequ int)sequ int:
   (* definiert nur für s ∈ S3 *)
   if isempty(s) then s
   else prefix(first(s),vorn(rest(lead(lead(s))))) endif
```

Beweisen Sie: Für $s \in S_3$ ist *vorn*(*s*) das "vordere Drittel von *s*", d.h. für $s = (s_1,\ldots,s_{3k})$, $k \in \mathbb{N}_0$, ist $vorn(s) = (s_1,\ldots,s_k)$.

c) Geben Sie analog zur Funktion *vorn* aus Teilaufgabe b) eine rekursive Funktionsvereinbarung für eine Funktion *hinten* an, die nur die genannten Grundoperationen und Funktionen aus Teilaufgabe a) verwendet und die für $s \in S_3$ das hintere Drittel von *s* berechnet (d.h. $hinten(s) = (s_{2k+1},\ldots,s_{3k})$ für $s = (s_1,\ldots,s_{3k})$, $k \in \mathbb{N}_0$).

d) Geben Sie eine rekursive Funktionsvereinbarung für eine Funktion *mitte* an, die außer den genannten Grundoperationen und den Funktionen aus Teilaufgabe a) ausschließlich die Funktion *vorn* aus Teilaufgabe b) verwenden darf und die für $s \in S_3$ das mittlere Drittel von *s* berechnet (d.h. $mitte(s) = (s_{k+1},\ldots,s_{2k})$ für $s = (s_1,\ldots,s_{3k})$, $k \in \mathbb{N}_0$).

e) Die rekursive Funktionsvereinbarung von *vorn* in Teilaufgabe b) soll in systematischer Weise in einen iterativen Algorithmus (mit gleicher Wirkung) überführt werden (der ebenfalls nur die angegebenen Grundoperationen und Funktionen aus Teilaufgabe a) verwendet). Betten Sie dazu *vorn* in einen geeigneten allgemeineren repetitiv rekursiven Algorithmus ein, und entrekursivieren und spezialisieren Sie diesen zu dem gesuchten iterativen Algorithmus.

f) Objekte aus **sequ** int können in Programmiersprachen wie PASCAL, MODULA o.ä. als lineare Listen realisiert werden. Geben Sie (in einer derartigen Programmiersprache) entsprechende Typvereinbarungen und Algorithmen zur Realisierung der angegebenen Grundoperationen an.

Fortsetzung nächste Seite!

g) Geben Sie (in PASCAL, MODULA o.ä.) einen iterativen Algorithmus *laenge* an, der unter Verwendung der Realisierungen der Grundoperationen gemäß Teilaufgabe f) die Anzahl der Komponenten einer als lineare Liste realisierten Folge aus **sequ** int berechnet.

Aufgabe 4

Durch die Funktionsvereinbarung

function $f(x,y,z$:nat)nat:
if $x = y+1$ **then** $z+y$ **else** $f(x+y+1, 2*(y+1), z+y+1)$ **endif**

ist eine Funktion $f: \mathbb{N}_0^3 \to \mathbb{N}_0$ definiert.

a) Bestimmen Sie $f(6,0,1)$.

b) Beweisen Sie: $f(x,y,z)$ terminiert für alle $x,y,z \in \mathbb{N}_0$ mit $x > y$.

c) Beweisen Sie, daß für alle $x,y,z \in \mathbb{N}_0$ mit $x > y$ gilt: $f(x,y,z)$ ist genau dann eine gerade Zahl, wenn $x + z$ ungerade ist.

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Frühjahr 1995

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 1995

66110

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie,Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 4

Bitte wenden!

Sämtliche Teilaufgaben sind zu bearbeiten!

Aufgabe 1

Gegeben sei die Grammatik Γ_1 mit {**a**,**b**} als Menge der Terminalzeichen, den Nichtterminalzeichen Z,A,B, dem Axiom Z und den Produktionsregeln

$$Z \rightarrow \mathbf{a}B$$
$$Z \rightarrow \mathbf{b}A$$
$$A \rightarrow \mathbf{a}$$
$$A \rightarrow \mathbf{a}Z$$
$$B \rightarrow \mathbf{b}$$
$$B \rightarrow \mathbf{b}Z$$

Die Grammatik Γ_2 entstehe aus Γ_1 dadurch, daß zu diesen Produktionsregeln noch die weiteren Regeln

$$A \rightarrow \mathbf{b}AA$$
$$B \rightarrow \mathbf{a}BB$$

hinzugenommen werden.

Der jeweilige Sprachschatz von Γ_1 und Γ_2 sei mit $\mathcal{L}(\Gamma_1)$ bzw. $\mathcal{L}(\Gamma_2)$ bezeichnet. ε bezeichne das leere Wort.

a) Beweisen Sie: $\mathcal{L}(\Gamma_1) = \{\mathbf{ab},\mathbf{ba}\}^* \setminus \{\varepsilon\}$.

b) Konstruieren Sie direkt aus Γ_1 einen nicht-deterministischen endlichen Automaten, der genau die Zeichenreihen von $\mathcal{L}(\Gamma_1)$ akzeptiert. Konstruieren Sie dann aus diesem Automaten einen deterministischen endlichen Automaten, der genau die Zeichenreihen von $\mathcal{L}(\Gamma_1)$ akzeptiert.

c) Geben Sie eine (deterministische) Turingmaschine T an, die außer einem Leerzeichen nur die Zeichen aus {**a**,**b**} verwendet und $\mathcal{L}(\Gamma_1)$ in folgendem Sinne akzeptiert: Ein Wort $w \in \{\mathbf{a},\mathbf{b}\}^*$ steht auf dem ansonsten leeren Band. Angesetzt auf das erste Zeichen von w (bzw. auf ein Leerzeichen, falls w das leere Wort ist), erreicht T genau dann nach endlich vielen Schritten einen Endzustand, wenn $w \in \mathcal{L}(\Gamma_1)$ ist.

d) Beweisen Sie: **aaabbabbba** $\in \mathcal{L}(\Gamma_2)$.

e) Für ein $w \in \{\mathbf{a},\mathbf{b}\}^*$ entstehe $\overline{w}$ aus w, indem man jedes **a** in w durch **b** und jedes **b** in w durch **a** ersetzt.

Beweisen Sie: Ist $w \in \mathcal{L}(\Gamma_2)$, so ist auch $\overline{w} \in \mathcal{L}(\Gamma_2)$.

f) Überführen Sie Γ_2 in Chomsky-Normalform.

Fortsetzung nächste Seite!

Aufgabe 2

NAT bezeichne den abstrakten Datentyp mit der Sorte nat der natürlichen Zahlen (einschließlich 0) und den üblichen Operationen auf nat. Der abstrakte Datentyp VEKTOR sei wie folgt definiert:

```
abstract type VEKTOR
  uses NAT                (* Alles, was NAT enthält, darf verwendet werden *)
  sorts vektor, index     (* Die Sorten von VEKTOR *)
  functions null: → vektor,
            proj: vektor × index → nat,
            succ: vektor × index → vektor
  axioms ∀x ∈ vektor ∀i,j ∈ index:
         proj(null,i) = 0,
         proj(succ(x,i),i) = proj(x,i) + 1,
         proj(succ(x,i),j) = proj(x,j)    für i ≠ j
endoftype
```

a) Die Funktion f: nat × index → vektor sei gegeben durch die Funktionsvereinbarung

```
function f(k:nat,i:index) vektor:
    if k = 0 then null else succ(f(k-1,i),i) endif
```

Beweisen Sie unter Verwendung der Axiome von VEKTOR, daß für alle $k \in$ nat und $i,j \in$ index gilt:

a1) $proj(f(k,i),i) = k$

a2) $proj(f(k,i),j) = 0$ für $i \neq j$

b) Geben Sie in Analogie zur Funktion f in Teilaufgabe a) eine rekursive Funktionsvereinbarung für eine Funktion g: vektor × nat × index → vektor an, für die für alle $k \in$ nat und $i,j \in$ index gilt:

b1) $proj(g(x,k,i),i) = proj(x,i) + k$

b2) $proj(g(x,k,i),j) = proj(x,j)$ für $i \neq j$

Beweisen Sie b1) und b2) für Ihre Lösung.

c) Für ein fest vorgegebenes $n \in \mathbb{N}$ sei nun index = $\{i \mid i \in \mathbb{N} \text{ und } 1 \leq i \leq n\}$ und vektor = $\{(x_1,...,x_n) \mid x_i \in \mathbb{N}_0 \text{ für } 1 \leq i \leq n\}$. Die Funktionen *null*, *proj* und *succ* seien gegeben durch

$null = (0,0,..,0)$ ("Nullvektor"),
$proj((x_1,...,x_n),i) = x_i$,
$succ((x_1,...,x_n),i) = (x_1,...,x_{i-1},x_i+1,x_{i+1},...,x_n)$.

c1) Zeigen Sie, daß die so definierten Funktionen die Axiome von VEKTOR erfüllen.

c2) Objekte der Sorte vektor können in höheren Programmiersprachen als Reihungen der Länge n realisiert werden (Typbezeichnung etwa **array** [1..n] **of** nat). Geben Sie (in der Notation einer derartigen Programmiersprache) Algorithmen zur Realisierung der Funktionen *null*, *proj* und *succ* an.

c3) Geben Sie (in einer Notation wie in Teilaufgabe c2)) einen Algorithmus an, der für $(x_1,...,x_n),(y_1,...,y_n) \in$ vektor den "Summenvektor" $(x_1+y_1,...,x_n+y_n)$ berechnet und dabei nur die Funktionen *null*, *proj* und *succ* verwendet. Erläutern Sie die wesentlichen Schritte des Algorithmus durch geeignete Kommentare.

Fortsetzung nächste Seite!

Aufgabe 3

Durch die Funktionsvereinbarung

```
function f(x,y,z:nat)nat:
  if z = 0 then x+y
  else if y = 0 then 1
       else f(f(x,y-1,z),x,z-1) endif
  endif
```

ist eine Funktion $f: \mathbb{N}_0^3 \to \mathbb{N}_0$ definiert. Ferner sei die Funktion $g: \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0$ mit

$$g(x,y) = \sum_{k=0}^{y} x^k$$

gegeben.

a) Bestimmen Sie $f(3,4,1)$.

b) Beweisen Sie: $f(x,y,z)$ terminiert für alle $x,y,z \in \mathbb{N}_0$.

c) Geben Sie (in der Notation einer höheren Programmiersprache) unter Verwendung eines Kellers als zusätzlicher "Hilfs"-Datenstruktur einen iterativen Algorithmus an, der $f(x,y,z)$ für beliebige $x,y,z \in \mathbb{N}_0$ berechnet. Erläutern Sie Idee und wesentliche Schritte Ihrer Lösung.

d) Beweisen Sie, daß für alle $x,y \in \mathbb{N}_0$ gilt: $f(x,y,2) = g(x,y)$.

e) Beweisen Sie, daß die Funktion g sich unter ausschließlicher Verwendung von Addition und Multiplikation (auf $\mathbb{N}_0$) sowie primitiver Rekursion definieren läßt.

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Frühjahr 1996

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 1996

66110

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie,Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 2

Bitte wenden!

Sämtliche Teilaufgaben sind zu bearbeiten!

1) Zeigen Sie, daß es zu jedem (nichtdeterministischen) endlichen erkennenden Automaten Asp mit spontanen Übergängen einen äquivalenten (nichtdet.) endlichen erkennenden Automaten Ansp ohne spontane Übergänge gibt!

   (Hinweis: Bei einem spontanen Übergang (z,ε,z') wird kein Symbol eingelesen.)

2) Konstruieren Sie einen linear beschränkten Automaten, der die Sprache L = {u§v |u ist Prefix von v} akzeptiert!

3) Zeigen Sie, daß es zu jeder nichtleeren, rekursiv aufzählbaren Menge A natürlicher Zahlen eine primitiv rekursive Funktion f gibt, deren Bildmenge A ist!

4) Für welche Sprachen-Klassen der Chomsky-Hierarchie ist das Wortproblem entscheidbar (Begründung!)?

5) Erklären und vergleichen Sie die Aufrufprinzipien 'call by value' und 'call by reference'!

6) Geben Sie eine Syntax-Graphen-Darstellung zur EBNF

   A = "x" | "(" B ")". B = AC. C = {"+"A}.

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Herbst 1996

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 1996

66110

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie,Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 3

Bitte wenden!

Sämtliche Teilaufgaben sind zu bearbeiten!

Aufgabe 1

Gegeben Sei folgendes Pascal-Programm:

```
program Parametertest;

 var n: integer;
 var a: array[1..2] of integer;

procedure update( x,y : integer );
var n: integer;
begin
  n:= 1;
  x:= x+n;
  y:= x*y;
end;

begin
n:=1;
a[1]:= 2;
a[2]:= 7;
update(n,a[n]);
end.
```

Geben Sie für die folgenden Parameterübergabetechniken jeweils an, welche Werte die Variablen n, $a[1]$ und $a[2]$ am Ende der Programmausführung haben.

(a) call-by-value

(b) call-by-reference

(c) call-by-name

Aufgabe 2

Gegeben sei eine Funktion $d : \mathbf{N}_\perp \times \mathbf{N}_\perp \to \mathbf{N}_\perp$, die wie folgt definiert ist:

$$d(x,y) =_{def} \begin{cases} \perp & \text{, falls } x = \perp \text{ oder } y = \perp \\ 1 & \text{, falls ein } z \text{ existiert mit } z * y = x \\ 0 & \text{, sonst} \end{cases}$$

(a) Beschreiben Sie informell (Stichworte), welche Funktion durch d dargestellt wird.

(b) Schreiben Sie eine rekursive Funktion, welche die Funktion d berechnet! Die Funktionen DIV (Division) und MOD (modulo) dürfen **nicht** verwendet werden.

(c) Stellen Sie das zu Ihrer Funktion gehörende Funktional Θ auf. Geben Sie dabei auch die Funktionalität von Θ an.

(d) Zeigen Sie, daß d ein Fixpunkt des Funktionals Θ ist, d.h. daß Ihr Programm eine Implementierung von d ist.

Fortsetzung nächste Seite!

Aufgabe 3

Programmieren Sie in Modula *oder* Pascal die folgenden Datentypen, Funktionen und Prozeduren:

(a) Definieren Sie rekursiv den Typ `Tree` der binären Bäume, deren Knoten ganze Zahlen enthalten.

(b) Programmieren Sie eine Funktion `is_in`, die einen binären Baum t aus Aufgabe (a) und eine ganze Zahl n als Eingabeparameter nimmt und einen boole'schen Wert als Ergebnis liefert, so daß `is_in(t,n)` den Wert `TRUE` liefert genau dann, wenn n in t vorkommt.

(c) Schreiben Sie eine rekursive Funktion `schachtelung`:
`PROCEDURE schachtelung(x:REAL; ug, og: REAL; eps:REAL): REAL` ,
die näherungsweise die Wurzel einer reellen Zahl nach der Methode der Intervallschachtelung berechnet, d.h. `x` ist die Zahl, deren Wurzel berechnet werden soll, `ug` und `og` sind die Unter- bzw. Obergrenze des gerade betrachteten Intervalls, `eps` ist die Genauigkeit, mit der die Wurzel berechnet werden soll. Die Rekursion soll abgebrochen werden, falls $|m^2 - x| <$ `eps`. Dabei sei `m` der Mittelwert des Intervalls $[ug \ldots og]$. Die Funktion wird aufgerufen mittels `schachtelung(x, 0.0, x, eps)`.
Der Absolutbetrag kann mit Hilfe einer Funktion `ABS(x)` berechnet werden.

Aufgabe 4

Gegeben sei der deterministische endliche Automat $M = (Q, \Sigma, \delta, q_0, F)$ mit $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $F = \{q_4\}$ und

$$\begin{array}{ll} \delta(q_0,a) = q_1 , & \delta(q_2,a) = q_4 \\ \delta(q_0,b) = q_2 , & \delta(q_2,b) = q_3 \\ \delta(q_1,a) = q_4 , & \delta(q_3,a) = q_4 \\ \delta(q_1,b) = q_3 , & \delta(q_3,b) = q_4 \\ \delta(q_4,a) = q_3 , & \delta(q_4,b) = q_3 \end{array}$$

(a) Zeichnen Sie den Automaten als Übergangsdiagramm.

(b) Berechnen Sie einen äquivalenten Automaten mit minimaler Anzahl von Zuständen.

(c) Geben Sie den Sprachschatz des Automaten an (ohne Beweis).

(d) Ist diese Sprache Typ-3 (regulär) ? (mit Begründung !)

(e) Zeigen Sie, daß folgende Sprache $L \subseteq \{a\}^*$ nicht Typ-3 (regulär) ist:
$$L = \{a^p \mid p = \textstyle\sum_{i=1}^{n} i \text{ für ein } n \in \mathbf{N}, n > 0\}$$

Aufgabe 5

Gegeben sei die Typ-2 Grammatik $G = (V, \Sigma, P, S)$ mit

$$V = \{S, A\}, \quad \Sigma = \{(,)\}, \quad P = \{ \begin{array}{llll} S \to (A, & S \to (AS, & & \\ S \to (SA, & S \to (SAS, & A \to) & \}. \end{array}$$

(a) Welche Sprache $\mathcal{L}(G)$ erzeugt G ?

(b) Geben Sie einen PDA (d.h. einen nichtdeterministischen Kellerautomaten) K an mit $N(K) = \mathcal{L}(G)$ (ohne Korrektheitsbeweis). Zur Erinnerung: $N(K)$ ist die Sprache, die von K durch leeren Keller erkannt wird.

(c) Inwiefern ist der PDA K geeignet zum Parsen der Sprache $\mathcal{L}(G)$? Welche Parsingtechniken kennen Sie ?

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Frühjahr 1997

Prüfungsteilnehmer	Prüfungstermin	Einze ·üfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 1997

66110

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie,Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 2

Sämtliche Teilaufgaben sind zu bearbeiten!

1. **Automatentheorie und Formale Sprachen**

   Zu jedem *nichtdeterministischen* endlichen Automaten (NFA) $\mathcal{A}$ kann man einen *deterministischen* endlichen Automaten (DFA) $\mathcal{A}'$ konstruieren, der die gleiche Sprache akzeptiert, d.h. $L(\mathcal{A}) = L(\mathcal{A}')$.

   (a) Geben Sie ein Verfahren für die Konstruktion von $\mathcal{A}'$ aus $\mathcal{A}$ detailliert an!

   (b) Für das Alphabet $\Sigma = \{a, b\}$ und natürliche Zahlen n seien Sprachen L_n und R_n definiert durch

   $$L_n := \Sigma^n \cdot b \cdot \Sigma^* \quad , \quad R_n := \Sigma^* \cdot b \cdot \Sigma^n$$

   Geben Sie NFAs $\mathcal{L}_n$ bzw. $\mathcal{R}_n$ an, welche L_n bzw. R_n akzeptieren!

   (c) Konstruieren Sie in den Fällen $n = 0, 1, 2$ die zugehörigen DFAs $\mathcal{L}'_n$ und $\mathcal{R}'_n$.

   (d) Zeigen Sie, dass der minimale DFA, der L_n akzeptiert, $n + 3$ Zustände hat.

   (e) Zeigen Sie, dass der minimale DFA, der R_n akzeptiert, 2^{n+1} Zustände hat.

Fortsetzung nächste Seite!

2. Berechenbarkeit

Zu den klassischen Problemen in der Theorie der Berechenbarkeit zählt das sog. "Korrespondenzproblem von POST" (PCP).

(a) Formulieren Sie die Problemstellung des PCP.

(b) Das PCP ist algorithmisch unentscheidbar. Begründen Sie dies!

(c) Die Unentscheidbarkeit des PCP kann dazu verwendet werden, mittels Reduktion die Unentscheidbarkeit von Problemen im Bereich der kontextfreien Grammatiken und Sprachen nachzuweisen. Geben Sie ein konkretes Beispiel hierfür an (Problemstellung und Reduktion)!

3. Algorithmische Sprachen

Zwei bekannte Analysealgorithmen für allgemeine kontextfreie Sprachen sind mit den Namen COOKE/YOUNGER/KASAMI (CYK) und EARLEY verbunden.

(a) Erläutern Sie die Vorgehensweise bei beiden Algorithmen in ihren Grundzügen.

(b) Welche Aussagen über die Laufzeitkomplexität (evtl. auch für Teilfamilien der Familie aller kontextfreien Sprachen) sind Ihnen bekannt?

(c) Stellen Sie einen der beiden Algorithmen detailliert dar (z.B. in Pseudocode), begründen Sie dessen Korrektheit und Laufzeitkomplexität.

4. Programmiermethodik, Effiziente Algorithmen

Eine der grundlegenden Aufgabenstellungen der Algorithmik ist die Identifizierung und Klassifikation von Problemen, für die es effiziente Entscheidungsalgorithmen bzw. effiziente Verifikationsalgorithmen gibt. Die entsprechenden Problemklassen werden mit $\mathcal{P}$ bzw. $\mathcal{NP}$ bezeichnet.

(a) Geben Sie Definitionen für die Klassen $\mathcal{P}$ und $\mathcal{NP}$ an und erläutern Sie diese an Beispielen.

(b) Erläutern Sie, in welchem Sinne die o.g. Klassen "robust" gegenüber definitorischen Variationen sind.

(c) Was versteht man unter der $\mathcal{P}$-vs.-$\mathcal{NP}$-Problematik?

(d) Beschreiben Sie das Reduktionskonzept innerhalb der Klasse $\mathcal{NP}$ an einem Beispiel.

(e) Was sind und welche Bedeutung haben $\mathcal{NP}$-vollständige Probleme? Erläutern Sie diese Begriffsbildung und geben Sie Beispiele für solche Probleme an.

(f) Betrachten Sie das Problem, bei dem man danach fragt, ob eine vorgelegte aussagenlogische Formel eine Tautologie ist, d.h. ob sie unter allen Belegungen der in ihr vorkommenden Aussagenvariablen mit Wahrheitswerten "wahr" bzw. "falsch" den Wert "wahr" annimmt.
Diskutieren Sie den Status dieses Problems in Bezug auf die Problemklassen $\mathcal{P}$ und $\mathcal{NP}$.

66110

Automatentheorie, Algorithmische Sprache (vertieft)

Herbst 1997

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 1997

66110

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: Informatik (vertieft studiert)

Einzelprüfung: Automatentheorie,Algorithm. Sprachen

Anzahl der gestellten Themen (Aufgaben): 1

Anzahl der Druckseiten dieser Vorlage: 4

Bitte wenden!

Sämtliche Teilaufgaben sind zu bearbeiten!

1. Aufgabe (Rechenstrukturen und Termersetzungssysteme)

Gegeben sei eine Rechenstruktur NAT mit der Signatur:

$$\Sigma = (S_{\mathrm{NAT}}, F_{\mathrm{NAT}}) \text{ mit } S_{\mathrm{NAT}} = \{\mathbf{bool}, \mathbf{nat}\} \text{ und } F_{\mathrm{NAT}} = \{\mathtt{zero}, \mathtt{succ}, \mathtt{pred}\}$$

Den beiden Sorten seien die folgenden Trägermengen zugeordnet:

$$\mathbf{bool}^{\mathrm{NAT}} = B^{\perp} := B \cup \{\perp\}, \quad \mathbf{nat}^{\mathrm{NAT}} = N^{\perp} := N \cup \{\perp\}$$

Dabei ist $N = \{0, 1, 2, \ldots\}$ die Menge der natürlichen Zahlen und $B = \{\mathrm{TRUE}, \mathrm{FALSE}\}$ die Menge der Booleschen Werte. Das Symbol $\perp$ steht für „undefiniert".

Die Spezifikationen der den Funktionssymbolen zero, succ und pred zugeordneten strikten Funktionen lauten:

$$\begin{array}{lll} \mathtt{zero}^{\mathrm{NAT}}: & \rightarrow N^{\perp}; & \mathtt{zero}^{\mathrm{NAT}} = 0; \\ \mathtt{succ}^{\mathrm{NAT}}: N^{\perp} & \rightarrow N^{\perp}; & \mathtt{succ}^{\mathrm{NAT}}(x) = x+1; \\ \mathtt{pred}^{\mathrm{NAT}}: N^{\perp} & \rightarrow N^{\perp}; & \mathtt{pred}^{\mathrm{NAT}}(x) = x-1 \text{; falls } x \geq 1 \text{ und } \mathtt{pred}^{\mathrm{NAT}}(0) = \perp \end{array}$$

Im folgenden nehmen wir an, daß jede natürliche Zahl durch einen Grundterm dargestellt ist, der nur aus den Konstruktoren zero und succ besteht.

a) Geben Sie ein Termersetzungssystem an, das die Funktion pred auf die Konstruktoren zero und succ zurückführt!

b) Nun sollen die strikten Funktionen add und sub mit den Spezifikationen

$$\begin{array}{lll} \mathtt{add}^{\mathrm{NAT}}: N^{\perp} \times N^{\perp} & \rightarrow N^{\perp}; & \mathtt{add}^{\mathrm{NAT}}(x,y) = x+y \\ \mathtt{sub}^{\mathrm{NAT}}: N^{\perp} \times N^{\perp} & \rightarrow N^{\perp}; & \mathtt{sub}^{\mathrm{NAT}}(x,y) = x-y \text{ falls } x \geq y \\ & & \mathtt{sub}^{\mathrm{NAT}}(x,y) = \perp \text{ falls } x < y \end{array}$$

zu F^{NAT} hinzugefügt werden ($x, y \neq \perp$). Geben Sie Termersetzungssysteme an, die add bzw. sub durch zero, succ und pred darstellen.

c) Stellen Sie analog zur Teilaufgabe b) je ein Termersetzungssystem für die folgenden, ebenfalls strikten Funktionen auf (mit $x, y \neq \perp$):

$$\begin{array}{lll} \mathtt{mult}^{\mathrm{NAT}}: N^{\perp} \times N^{\perp} & \rightarrow N^{\perp}; & \mathtt{mult}^{\mathrm{NAT}}(x,y) = x \cdot y \\ \mathtt{div}^{\mathrm{NAT}}: N^{\perp} \times N^{\perp} & \rightarrow N^{\perp}; & \mathtt{div}^{\mathrm{NAT}}(x,y) = x \div y \text{ falls } y > 0 \\ & & \mathtt{div}^{\mathrm{NAT}}(x,0) = \perp \end{array}$$

Sie dürfen dabei auch die Funktionen add, sub aus b) verwenden.

Wir betrachten nun den seit dem Altertum bekannten Euklidischen Algorithmus in der Notation einer imperativen Programmiersprache (a, b seien von der Sorte **nat**)

(*)
```
while a ≠ b do
    while a < b do b := b − a od
    (a, b) := (b, a)
od
```

Fortsetzung nächste Seite!

d) Stellen Sie einen exemplarischen Ablauf des Algorithmus für die Variablenbelegung $a = 32, b = 18$ als Folge von Zuständen des Variablenraumes dar.

e) Der Algorithmus soll nun mit Hilfe der Zusicherungsmethode nach Floyd und Hoare verifiziert werden. Geben Sie für beide `while`–Schleifen geeignete Invarianten an, und beweisen Sie damit die Korrektheit des Algorithmus.
Hinweis: Gehen Sie von der Zusicherung $\{P \wedge a = mg \wedge b = ng \wedge n, m \text{ teilerfremd}\}$ mit $P = a > 0 \wedge b > 0$ vor Beginn der ersten `while`–Anweisung aus.

f) Formulieren Sie den Euklidischen Algorithmus rekursiv in einer beliebigen Notation.

g) Geben Sie ein Termersetzungssystem auf der Rechenstruktur NAT für den Euklidischen Algorithmus an. Sie dürfen dabei alle in den Teilaufgaben a) mit c) auf NAT eingeführten Funktionen und Hilfsfunktionen verwenden.

h) Beweisen Sie die Terminierung der beiden `while`–Schleifen in unserer ursprünglichen Formulierung (*) des Euklidischen Algorithmus.

2. Aufgabe (Rekursive Rechenstrukturen)

Gegeben sei die rekursive Rechenstruktur

$$Liste = Leer \mid (Float, Liste)$$

der Listen über reellen Gleitpunktzahlen mit den Funktionen

- $head:$ $Liste \rightarrow Float$
- $tail:$ $Liste \rightarrow Liste$
- $mklist:$ $Float \times Liste \rightarrow Liste$

Für diese Funktionen gelten folgende Spezifikationen:

- $head(x)$ und $tail(x)$ sind partiell nur für nicht–leere Listen definiert, und
  $head(x)$ ist das erste Element der Liste x,
  $tail(x)$ ist die um das erste Element verkürzte Liste x.
- $mklist(r, x)$ ist total und liefert die Liste, die durch Voransetzen des Elements r vor die Liste x entsteht.

Die Rechenstruktur *Liste* soll nun um die folgenden Funktionen erweitert werden:

- $length : Liste \rightarrow Int$
  mit der Spezifikation: $length(x)$ ist total und liefert die Anzahl der Elemente in der Liste x.

Fortsetzung nächste Seite!

- $proj : Int \times Liste \rightarrow Float$
  mit der Spezifikation: $proj(n, x)$ ist partiell nur für nicht–leere Listen x sowie für n mit $1 \leq n \leq length(x)$ definiert und liefert das n–te Element der Liste x.

- $part : Int \times Int \times Liste \rightarrow Liste$
  mit der Spezifikation: $part(m, n, x)$ ist partiell nur für nicht–leere Listen x sowie für m und n mit $1 \leq m \leq n \leq length(x)$ definiert und liefert die Teilliste von x vom m–ten bis zum n–ten Element einschließlich.

1. Programmieren Sie die 3 Funktionen *length*, *proj* und *part* rekursiv unter Abstützung auf die primitiven Rechenstrukturen *Int* und *Bool* sowie auf die oben angegebene Rechenstruktur *Liste*.

2. Beweisen Sie für das von Ihnen für die Funktion *part* angegebene Programm,
   (a) daß es für alle zulässigen Parameter terminiert und
   (b) daß es die Spezifikation für die Funktion *part* erfüllt.

3. Aufgabe (Endliche Automaten und reguläre Mengen)

Gegeben sei das Alphabet $\mathcal{A} = (A, B, C)$. In $\mathcal{A}^*$ zeichnen wir die Teilmenge $\mathbf{T}$ der Wörter aus, die weder ACC noch BCC als Teilzeichenreihe enthalten. Dabei ist $x \in \mathcal{A}^*$ genau dann eine Teilzeichenreihe von $y \in \mathcal{A}^*$, wenn es ein $y' \in \mathcal{A}^*$ und ein $y'' \in \mathcal{A}^*$ gibt, so daß $y = y'xy''$ ist.

1. Konstruieren Sie den (bis auf die Bezeichnungen der Zustände eindeutigen) minimalen deterministischen endlichen Automaten $\mathbf{A} = (S, I, \delta, s_0, F)$ mit der Zustandsmenge S, dem Eingabealphabet $I = \mathcal{A}$, der Zustandsübergangsfunktion $\delta : S \times I \rightarrow S$, dem Anfangszustand s_0 und der Endzustandsmenge F, der genau $\mathbf{T}$ akzeptiert! Stellen Sie hierzu den Automaten $\mathbf{A}$ durch seinen Zustandsübergangsgraphen dar.

2. Beweisen Sie, daß der von Ihnen in der Antwort zu 1. angegebene Automat $\mathbf{A}$
   (a) genau $\mathbf{T}$ akzeptiert und
   (b) minimal ist.

3. Stellen Sie $\mathbf{T}$ als eine reguläre Menge über $\mathcal{A}$ dar.