Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst

1998

66112

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

- Prüfungsaufgaben -

Fach: **Informatik (vertieft studiert)**

Einzelprüfung: **Automatentheorie,Komplexität,Algorith.**

Anzahl der gestellten Themen (Aufgaben): 2

Anzahl der Druckseiten dieser Vorlage: 8

Bitte wenden!

Thema Nr. 1

Sämtliche Teilaufgaben sind zu bearbeiten!

1. Grammatiken und Automaten

Wir wollen Boolesche Ausdrücke BA als Sprache über L* mit L = {0, 1, x, y, +, -, (,)} darstellen. Dabei stehen 0,1 für die Booleschen Konstanten, x, y für Boolesche Variable, + für das Disjunktionszeichen und ein vorausgestelltes - für die Negation. Das Konjunktionszeichen wird weggelassen. Die Klammern "(" und ")" können weggelassen werden, falls sie wegen der Regel, daß die Konjunktion stärker bindet als die Disjunktion, oder wegen der Assoziativgesetze überflüssig sind.

Unter Verwendung der genannten Symbole sollen die Booleschen Ausdrücke nach den folgenden Regeln aufgebaut werden:

i) Literale sind Boolesche Konstanten und Variablen. Ein negiertes Literal ist auch ein Literal. Nichts sonst ist ein Literal. Beispiel: - - - 0

ii) Schreibt man zwei BA hintereinander, so erhält man eine Konjunktion. Beispiel: x-1, xy0, 0y-1x.

iii) Jeder BA ist eine (triviale) Disjunktion. Die Summe zweier Disjunktionen ist wieder eine Disjunktion. Beispiel: x-1, x+1, y+-x+0.

iv) Jedes Literal und jede Konjunktion ist ein BA. Jede eingeklammerte Disjunktion ist ein BA.

a) Geben Sie eine kontextfreie Grammatik (Chomsky-Typ 2) für die oben definierten BA an und leiten Sie die beiden Ausdrücke 0x-1y und ((0+x)y-x) mit Hilfe Ihrer Grammatik ab.

b) Beweisen Sie die folgende Aussage:
Ersetzt man in der Sprache (bzw. der in Aufgabe I konstruierten Grammatik) alle Terminalzeichen bis auf die Klammern")" und "(" durch das leere Wort ε, dann ist die so erhaltene Sprache immer noch nicht regulär.

Nun definieren wir mit den obigen Gesetzen eine andere Sprache BA*, indem wir i), ii), iv) beibehalten und iii) folgendermaßen verändern:

iii)* Jedes Literal ist eine (triviale) Disjunktion. Die Summe zweier Disjunktionen ist wieder eine Disjunktion.

c) Stellen Sie eine Grammatik auf, die BA* erzeugt und leiten Sie damit die Ausdrücke (0)(x) und (1+x+y) ab.

d) Zeigen Sie, dass BA* regulär (vom Chomsky-Typ 3) ist, indem Sie einen endlichen Automaten angeben, der BA* erkennt.

e) Nun vereinfachen wir die Sprache BA aus Teilaufgabe a) zu BA', indem wir als einziges Literal das Zeichen *l* zulassen. Geben Sie einen Kellerautomaten an, der die Sprache BA' erkennt.

Fortsetzung nächste Seite!

2. Turingmaschinen und Berechenbarkeit

Wir betrachten eine Turingmaschine $T = (I, B, Q, \delta, q_0)$, wobei $I = \{0, 1\}$ das Eingabealphabet, $B = I \cup \{\#\}$ das Bandalphabet mit dem Leerzeichen #, $Q = \{q_0, .. ,q_6\}$ eine Menge von Zuständen, $\delta = Q \times B \rightarrow Q \times B \times \{\leftarrow, \downarrow, \rightarrow\}$ die Zustandsübergangsfunktion und q_0 der Anfangszustand ist.

Die Zeichen $\leftarrow$ bzw. $\rightarrow$ symbolisieren eine Bewegung des Schreib-/ Lesekopfes (nach der aktuellen Operation) nach links bzw. rechts. Die Maschine hält nach der aktuellen Operation, wenn die Zustandsübergangsfunktion nicht definiert ist oder auf ein $\downarrow$ führt.

Als Eingabewörter lassen wir nur Zeichenfolgen der Form 0^n10^m mit $n, m \in N_0$ zu, wobei 0^n für eine Zeichenkette aus n Nullen stehe. Am Anfang stehe der Schreib-/ Lesekopf auf dem ersten (linken) Zeichen des Eingabewortes. Links und rechts vom Eingabewort stehen ausschließlich Leerzeichen auf dem Band.

Die Werte der Zustands- Übergangsfunktion $\delta(q, b)$ seien durch die folgende Tabelle definiert:

Bandzeichen $b \in B$ Zustand $q \in Q$	0	1	#
q_0	$(q_1, \#, \rightarrow)$	$(q_5, \#, \rightarrow)$	$(q_6, \#, \downarrow)$
q_1	$(q_1, 0, \rightarrow)$	$(q_2, 1, \rightarrow)$	nicht definiert
q_2	$(q_3, 1, \leftarrow)$	$(q_2, 1, \rightarrow)$	$(q_4, \#, \leftarrow)$
q_3	$(q_3, 0, \leftarrow)$	$(q_3, 1, \leftarrow)$	$(q_0, \#, \rightarrow)$
q_4	$(q_4, 0, \leftarrow)$	$(q_4, \#, \leftarrow)$	$(q_6, 0, \downarrow)$
q_5	$(q_5, \#, \rightarrow)$	$(q_5, \#, \rightarrow)$	$(q_6, \#, \downarrow)$

a) Beschreiben Sie die vollständige Berechnung von T für das Eingabewort 0010, indem Sie für jeden Berechnungsschritt den jeweiligen Maschinenzustand und die jeweilige Bandbelegung angeben.

b) Stellen Sie die Arbeitsweise der Maschine T mit Hilfe eines geeigneten Zustandsübergangsdiagramms dar.

c) Die zulässigen Eingabewörter 0^n10^m sollen nun jeweils ein Paar natürlicher Zahlen (m, n) repräsentieren. Welche Funktion $f: N_0 \times N_0 \rightarrow N_0$ wird von T berechnet ? Begründen Sie Ihre Aussage.

d) Untersuchen Sie, ob Ihre in Teilaufgabe c) angegebene Funktion primitiv rekursiv ist und beweisen Sie Ihre Aussage.

e) Geben Sie eine möglichst kleine obere Schranke für die Anzahl der Berechnungsschritte von T für ein zulässiges Eingabewort der Länge $|w| = k$ an.

Fortsetzung nächste Seite!

3. Spezifikation und Implementierung

Gegeben sei die folgende Spezifikation der natürlichen Zahlen

NAT = **based_on** BOOL
sorts nat
ops 0: nat
s: nat → nat
p: nat → nat
+: nat × nat → nat
zero: nat → bool
axioms
nat **generated_by** 0, s
p(0) = 0
p(s(x)) = x
0 + y = y
s(x) + y = s(x+y)
zero(0) = True
zero(s(x)) = False

Dabei bedeutet **based_on** BOOL, daß NAT auf einer geeigneten Spezifikation von BOOL aufbaut. Die Aussage nat **generated_by** 0,s drückt aus, daß alle Elemente aus nat als Terme der Form s(s(..(0)..)) dargestellt werden können.

a) Nun soll die Spezifikation NAT in einer funktionalen Programmiersprache Ihrer Wahl implementiert werden, wobei natürliche Zahlen als Bitlisten, das heißt als Sequenzen Boolescher Werte vom Typ [bool] dargestellt werden sollen. Programmieren Sie die dazu nötigen Funktionen.

b) Geben Sie eine Abstraktionsfunktion phi: [bool] → Nat an, die jede Bitliste, die sich mit Hilfe Ihrer Implementierung erzeugen läßt, auf die dazugehörige natürliche Zahl abbildet. Mit Nat ist dabei die Sorte gemeint, die in der von Ihnen gewählten Programmiersprache die natürlichen Zahlen bzw. eine geeignete Obermenge davon implementiert.

Thema Nr. 2

Sämtliche Teilaufgaben sind zu bearbeiten!

1. a) Gegeben sei die Grammatik G mit {1,2,a,b} als Menge der Terminalzeichen, der Variablen S, der Startvariablen S und den Produktionen

$$S \to 1Sab,\ S \to lab,\ S \to 2Sbbb,\ S \to 2bbb.$$

Sei $L = L(G)$ die von G erzeugte Sprache, $w_1 = ab$ und $w_2 = bbb$.

A. Beweisen Sie $L(G) = \{i1 \ldots ik\ w_{ik} \ldots w_{i1} \mid i1 \ldots ik \in \{1,2\}\}$.

B. Konstruieren Sie einen deterministischen Keller-Automaten, der L akzeptiert.

C. Ist L regulär? (Begründung)

b) Beweisen oder widerlegen Sie für beliebige Sprachen L_1, L_2:

A. $(L_1 \cup L_2)^* = L_1^* \cup L_2^*$

B. $(L_1^*)^* = L_1^*$

2. Beweisen Sie: Ist $f : \mathbf{N} \to \mathbf{N}$ LOOP-berechenbar, so ist auch $g : \mathbf{N} \to \mathbf{N}$ mit

$$g(n) = \sum_{i=1}^{n} f(i)$$

LOOP-berechenbar.

3. 22 Vertreter aus sieben Staaten besuchen eine internationale Konferenz. Welche der folgenden Aussagen muss wahr sein? (Begründung)

a) Sechs Staaten haben jeweils drei Vertreter und der 7. Staat hat vier Vertreter.

b) Kein Staat hat mehr als vier Vertreter.

c) Mindestens ein Staat hat vier oder mehr Vertreter.

d) Kein Staat hat weniger als zwei Vertreter.

Fortsetzung nächste Seite!

4. a) (Aussagenlogik)

A. Beweisen Sie: " $(A \rightarrow B) \leftrightarrow (\neg A \vee B)$ " ist eine Tautologie.

B. Begründen Sie die Aussage:

- "Aus {F,G} folgt H. " ist äquivalent zu " $\{F, G, \neg H\}$ ist unerfüllbar."

C. Beweisen Sie mit dem Resolutionskalkül:

- $\{\neg A \vee B,\ \neg B \vee C,\ A,\ \neg C\}$ ist unerfüllbar.

D. Beweisen Sie (unter Verwendung der Teilaufgaben A., B. und C.):

- "Aus $\{A \rightarrow B,\ B \rightarrow C\}$ folgt $A \rightarrow C$. "

b. (Prädikatenlogik) Gegeben ist die folgende Formel:

- $F : ((\forall x\, \exists y\, P(x, y)) \rightarrow (\exists x\, \forall y\, P(x, y)))$

A. Ist F allgemeingültig?

B. Ist F erfüllbar?

C. Falls F ein Modell hat, geben Sie ein solches an.

5. a) Stellen Sie sich vor, Sie starten für jede der folgenden Teilaufgaben den SCHEME-Interpreter erneut und geben die folgenden Ausdrücke ein. Geben Sie für jede Teilaufgabe das Ergebnis der Auswertung des letzten Ausdrucks an.

i)
```
(define zwei 2)
(define drei 3)
(define eins 8)
(define plus +)
(define kleiner <)
(kleiner eins (plus zwei drei))
```

ii)
```
(cons 'a (cons 'b (cons (cdr (cons (cons 'd 1) (cons 'c 2))) '())))
```

iii)
```
(define (rek a)
    (cond ((= a 0) 5)
          ((= a 5) (* 2 (rek (- a 5))))
          (else (lambda (x) (rek (- x 5))))))
((rek (rek 5)) (rek 5))
```

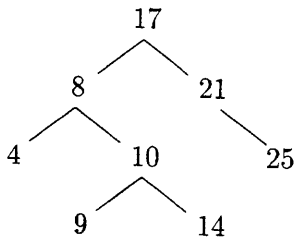
Fortsetzung nächste Seite!

b) Definieren Sie in einer funktionalen Sprache Ihrer Wahl eine Funktion, die das Skalarprodukt zweier Vektoren berechnet, die in rechtwinkligen Koordinaten gegeben sind. Die Vektoren sollen als Listen repräsentiert sein. Sie können davon ausgehen, dass Ihre Funktion nur mit Vektoren aufgerufen wird, die gleichlang sind.
Beispiel in SCHEME: (skalarprodukt ' (1 2 3) '(2 2 2)) liefert 12.

6. Ein geordneter binärer Baum ist entweder ein leerer Baum oder besteht aus einer Wurzel und zwei binären Bäumen als linkem und rechtem Unterbaum. Dabei sind die Knoten des Baums so geordnet, dass für alle Knoten v' im linken Unterbaum und für alle Knoten v'' im rechten Unterbaum von Knoten v gilt:

$$v' \leq v \leq v''$$

In SCHEME werden Bäume durch Listen kodiert. Der folgende Baum



hat die Listendarstellung:

```
'(17 (8 (4 ( ) ( ))
        (10 (9 ( ) ( ))
            (14 ( ) ( ))))
    (21 ( )
        (25 ( ) ( ))))
```

a) Definieren Sie in einer funktionalen Sprache Ihrer Wahl die folgenden Funktionen:

* root liefert den Wert an der Wurzel des Baumes.
* left -branch liefert den linken Teilbaum der Wurzel.
* right-branch liefert den rechten Teilbaum der Wurzel.
* make-tree bildet aus einer Wurzel und zwei Bäumen einen neuen Baum.
* Der leere Baum sei als leere Liste definiert. Definieren Sie die Funktion empty-tree?, die abfragt, ob ein Baum leer ist.

b) Definieren Sie die Funktion contains?, die als Argumente einen binären Baum und ein Element nimmt und überprüft, ob das Element im Baum enthalten ist.

c) Definieren Sie die Funktion insert, die als Argumente einen binären Baum und ein Element nimmt und das Element an der richtigen Stelle im Baum einsortiert.

Fortsetzung nächste Seite!

7. Für Ihr örtliches Rathaus sollen Sie eine Datenbank entwerfen, die personenbezogene Daten zu jedem Einwohner Ihrer Stadt speichert. Zu jedem Einwohner und jeder Einwohnerin werden die personenbezogenen Daten gespeichert, die auf seinem bzw. ihrem Personalausweis vermerkt sind. Diese sind eine 9-stellige Personalausweisnummer, Vor- und Nachname der Person, Geburtsdatum, Geburtsort, Staatsangehörigkeit, Ablaufdatum der Gültigkeit, Wohnort, Augenfarbe und Größe.

   a) Definieren Sie mit den Mitteln einer beliebigen imperativen Programmiersprache (z.B. C) zunächst einen geeigneten Datentyp pers_Daten, der die oben genannten Informationen aufnehmen soll. Sie können dabei davon ausgehen, dass kein Orts- oder Personenname länger als 40 Zeichen ist.

   b) Sie wollen die Personen nun in Stadtteilkarteien zusammenstellen. Ein Stadtteil besteht aus einer konstanten maximalen Anzahl von Einwohnern. Definieren Sie mit den Mitteln der gewählten Programmmiersprache zunächst eine Konstante MAXANZAHL mit dem Wert 12000. Legen Sie anschließend einen Datentyp stadtteil fest, der eine feste, vorher bekannte Anzahl von Einwohnern, nämlich MAXANZAHL, vom Typ pers_Daten aufnehmen soll.