Interstaatliche Hochschule für Technik Buchs NTB, Sy	Diskrete Mathematik
Block Informatik	9.12.2003
Serie 12	**Test**

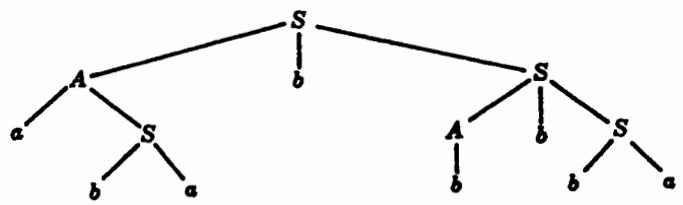
Name	Vorname	54 Pt. total (34) →	Note

– Punkteabzug oder keine Punkte für schwer verständliche oder unnötig komplizierte Lösungen.
– Hilfsmittel: ein A4-Blatt, beidseitig, selber geschrieben.
– Zeit: 80 Minuten.

Aufgabe 1 (1+1+1+2+1+1+2 Pt.)

Pt. 9 (6)

Sie sehen einen Herleitungsbaum (*parse tree*):



(a) Wie lautet das hergeleitete Wort?

abababbba (1)

(b) Welches sind die *terminals*?

a, b (1)

(c) Welches sind die *nonterminals*?

S, A (1)

(d) Welche Produktionsregeln wurden für diesen *parse tree* verwendet?

$S \to AbS$ $\quad A \to aS$
$S \to ba$ $\quad A \to b$

– nur kleine Flüchtigkeits-fehler: – 1 Pt. (2)

(e) Von welchem Typ sind diese Produktionsregeln?

Typ 2 : kontext-frei (1)

(f) Nennen Sie ein weiteres – möglichst einfaches – Wort der Sprache.

ba (1)

(g) Gibt es ein Wort der Länge 100 (genau 100!) in dieser Sprache?
Falls ja, dann nennen Sie ein möglichst einfaches, falls nein, warum?

z.B. $\underbrace{bb\ldots b}_{99}a = b^{99}a$

– nicht das Einfachste, aber korrekt: 1 Pt.
– muss als Lsg. jedoch einleuchtend sein, sonst 0 Pt. (2)

Allgemein:
– bis auch (d) + (g) nur 50% Punkte (richtig).
– (g) nur Punkte, falls (d) korrekt!
– (e) + (f) bei Fehler (d) nur Pt. falls nicht vereinfacht!

Aufgabe 2 (11x 1 Pt.)

11 Pt.

8

(a) `TYPE Vector = ARRAY 3 OF REAL;` (das ist Komponenten-Pascal ;-)

Diese Struktur ist: ☒ endlich ☐ abzählbar (unendlich) ☐ überabzählbar (1)

(b) Menge der ∞-langen ASCII-Strings

Diese Struktur ist: ☐ endlich ☐ abzählbar (unendlich) ☒ überabzählbar (1)

(c) Die reellen Zahlen zwischen 0.538 und 0.539

Diese Struktur ist: ☐ endlich ☐ abzählbar (unendlich) ☒ überabzählbar (1)

(d) Die Menge der geraden Zahlen

Diese Struktur ist: ☐ endlich ☒ abzählbar (unendlich) ☐ überabzählbar (1)

(e) Menge der Namen aus höchstens 20 Buchstaben: {Urs, Olaf, Emmenegger, ...}

Diese Struktur ist: ☒ endlich ☐ abzählbar (unendlich) ☐ überabzählbar (1)

(f) Was versteht man unter einer *metalanguage*?

Eine Sprache, welche eine andere Sprache beschreibt. (1)

(g) Gibt es einen Unterschied zwischen *nonterminals* und *syntactic categories*? Wenn ja, welchen?

Nein (1)

(h) Von welchem Typ ist die Grammatik $\{S \to aAB,\ S \to AB,\ A \to a,\ B \to b\}$?

Typ 2: Kontext-frei (1)

(i) Von welchem Typ ist die Grammatik $\{S \to aB,\ B \to AB,\ \underline{aA \to b},\ A \to b\}$?

Typ 0: allgemein (1)

(j) Von welchem Typ ist die Grammatik $\{S \to aB,\ B \to bB,\ B \to bA,\ A \to a,\ B \to b\}$?

Typ 3: regulär (1)

(k) Von welchem Typ ist die Grammatik $\{S \to Bb,\ B \to a,\ \underline{a \to B},\ B \to a\}$?

Ist keine Grammatik (1)

Aufgabe 3 (4x3 Pt.)

12 Pt.

8

(a) Welche Sprache erzeugt folgende Grammatik?

$$\{S \to aB,\ B \to b,\ B \to bA,\ A \to aB,\ A \to bC,\ C \to Ca\}$$

$L = \{ (ab)^n \mid n > 0 \}$

- $a^n b^n$: 2 Pt.
- $n \geq 0$: 2 Pt.

3

(b) Kann die Sprache von Teilaufgabe a durch einen regulären Ausdruck erzeugt werden? Wenn ja, durch welchen, wenn nein, warum nicht?

$ab(ab)^*$

3

- nur $(ab)^*$: 1 Pt.

(c) Schreiben Sie die Grammatik $\{S \to aB,\ B \to bB,\ B \to bA,\ A \to a,\ B \to b\}$ in BNF.

<S> ::= a <B>
<A> ::= a
<B> ::= b<B> | b<A> | b

3

- Syntax nicht 100% BNF : 2 Pt.
- EBNF : 1 Pt.

(d) Lassen sich bei jedem Sprachen-Typ (0-3) *derivation trees* zeichnen?
(Mit sauberer Begründung)

Nein, nur Typ 2 und Typ 3 (rechts) +1
weil bei jedem Knoten mit Söhnen
genau ein Nonterminal stehen muss.

- Begründung mangelhaft: - 1 Pt.

3

Aufgabe 4 (4+3+3+4 Pt.)

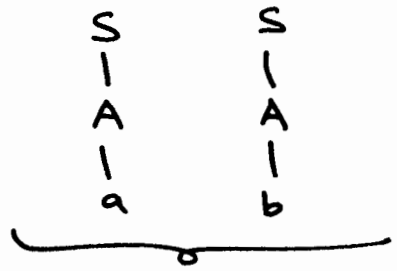
14 Pt.

7

(a) Notieren Sie eine einfache Grammatik, die auf verschiedene Arten geparst werden kann. Zeigen Sie von einem Wort der Sprache, dass dieses verschiedene *derivation trees* hat.

z.B. $S = A \mid B$
$A = a$
$B = a$

Für das Wort a:



zwei verschiedene Herleitungsbäume!

- Bsp. darf kompliziert sein.
- Grammatik zu kompliziert: -1/-2 Pt.
- Grammatik eindeutig zum Parsen: 0 Pt.

(4)

Finden Sie reguläre Ausdrücke zu folgenden Sprachen in a und b:

(b) Die Worte beginnen mit aa und enden mit einer beliebigen Anzahl von b's.

das ist hier der Fall

$aa(a|b)^*$

- mit "b*" am Ende: -1 Pt.
- aab^*: 1 Pt.

(3)

(c) Die Worte beginnen und enden mit b, und nirgends kommt bab vor.

d.h. a kommt nie einzeln vor

$b(aaa^*|b)^*b$

- nur einfache Gr: 3 Pt.
- kleine Fehler: 2 Pt.
- Ansätze: 1 Pt.

(3)

Konstruieren Sie eine reguläre Grammatik:

(d) Sei L die Sprache aller Worte in a und b, mit einer geraden Anzahl von a's. (Tipp: Diese Aufgabe ist mit 5 Regeln zu schaffen!)

S: # a's ist gerade
A: # a's ist ungerade

$S \to \lambda \quad S \to bS \quad S \to aA \quad A \to bA \quad A \to aS$

- nicht regulär: 0 Pt.
- kleine Fehler: -1 Pt.

(4)

Aufgabe 5 (4+4 Pt.)

Pt. 8

5

(a) Schreiben Sie in *Java* die Methode zum Parsen der folgenden Zeile EBNF. Die Methodem zum Einlesen von Zeichen, Parsen von `Factor`, Fehlerbehandlung, etc. sind bereits vorhanden. Bitte sauberes Top-Down-Parser Design, so wie in der Stunde gelernt!

```
Term  = Factor | Factor ("*"|"/") Factor
```

```
void Term() {
    Factor();
    if (ch=='*' || ch=='/') {
        nextchar();
        Factor();
    }
}
```

- nur 100% richtig : 4 Pt.
- Detailfehler, unwichtig : 3 Pt.
- Fehler, aber Struktur korrekt : 2 Pt.
- allgemeine Struktur : 1 Pt.

4

(b) Wie lautet Ihre Methode von Teilaufgabe a, wenn zusätzlich der *Parse-Tree* im Speicher erzeugt werden soll? Notieren Sie bitte auch die notwendige Klassen-Deklaration zur Definition des Baums. Bitte sauberes Design, so wie in der Stunde gelernt!

```
Tree Term() {
  Tree t = new Tree();
  t.type = 1; // Term = 1
  t.left = Factor();
  if (ch=='*' || ch=='/') {
      t.op = ch;
      nextchar();
      t.right = Factor();
  }
  return t;
}
```

```
class Tree {
   byte type;
   char op;
   Tree left, right;
}
```

- Bewertung wie Teilaufgabe a.

4