technische universität dortmund

fakultät für informatik

Martin Apel Informatik VI, GB5/R 431; Tel.: 4775
Nicola Beume Informatik XI, OH14/R 233; Tel.: 7704
Jürgen Mäter Informatik IV, GB5/R 402; Tel.: 2411

Dortmund, den 26. November 2010

Übungen zur Vorlesung EINI (WS 2010)

Blatt 7

Block rot

Es können 4 + 3 Punkte erreicht werden.

Abgabedatum: 5. Dezember 2010 23:59 Uhr

Hinweise

1) Bitte beachten Sie die aktuellen Hinweise unter

   http://www4.cs.uni-dortmund.de/home/maeter/UeEiniET10W/

   Für die Abgabe sind die Dateien: *Aufgabe_07_0.txt*, *Aufgabe_07_1a.txt*, *Aufgabe_07_1b.cpp*, *Aufgabe_07_1d.cpp*, *Aufgabe_07_2a.txt*, *Aufgabe_07_2b.txt* und *Aufgabe_07_2c.txt* zu erstellen.
2) Es ist ratsam, die Programme vor der Abgabe zu kompilieren und auszuführen.
3) Verwenden Sie für die Textaufgaben reine Texteditoren. Das Abgabesystem erkennt **keine** Word- oder PDF-Dateien!!!
4) In der optionalen Datei *Anmerkungen.txt* können Sie allgemeine Anmerkungen bezüglich Ihrer Lösungen notieren.
5) Machen Sie sich im Vorhinein Gedanken, wie Sie die zu erstellenden Programme zu implementieren sind. Machen Sie sich also zuerst einen Programmentwurf.
6) Kommentieren Sie Ihre Quelltexte. Das gehört zur Aufgabenstellung und wird mitbewertet.

Spruch des Tages

(frei nach Murphy)

Glücklich sind die Benutzer, die nichts erwarten. Sie werden nicht enttäuscht.

Aufgaben

Aufgabe 0: Grundlagen (1 Punkt)

Legen Sie für Ihre Antworten eine Text-Datei *Aufgabe_07_0.txt* an.

a) Wie ist ein deterministischer endlicher Automat (DEA) definiert (0,5 Punkte) ?

b) Wie wird ein DEA ohne Ausgaben genannt (0,1 Punkt) ?

c) Geben Sie zu jeder der folgenden Funktionen die einzubindende Header-Datei (`#include <???.h>`) an: `sqrt()`, `rand()`, `isdigit()` und `time()` (0,4 Punkte) .

Speichern Sie Ihre Ergebnisse in der Ergebnisdatei *Aufgabe_07_0.txt*.

Aufgabe 1: DEA (3 Punkte)

Hinweis: Bei dieser Aufgabe handelt es sich um eine leicht modifizierte Prüfungsaufgabe aus dem Jahr 2008.

a) Entwerfen Sie einen deterministischen endlichen „Frosch"-Automaten $A = (S, \Sigma, \delta, F, s_0)$, der sogenannte „Frosch"-Strings akzeptiert. Ein „Frosch"-String sei dabei wie folgt definiert:

- Am Anfang steht das Zeichen `q`.
- Danach können beliebig viele Zeichen `u` folgen.
- Danach können beliebig viele Zeichen `a` folgen.
- Danach folgt ein Zeichen `c`, gefolgt von einem Zeichen `k`.
- Danach ist entweder Ende oder es folgt ein Zeichen `_` dem wieder ein „Frosch"-String folgt.

Beispiele für nach dieser Definition „gültige" „Frosch"-Strings wären `quack` und `quuack` oder `quuuack_quaaack`. Nach dieser Definition „ungültiger" „Frosch"-Strings wäre z.B. `quacke`.

Legen Sie eine Datei *Aufgabe_07_1a.txt* an und schreiben Sie analog zur Vorlesung die Übergangsfunktion δ Ihres Automaten A in Form einer Übergangsmatrix in diese Datei. „Testen" Sie Ihren Automaten anhand mehrerer Beispiele (1 Punkt).

b) Legen Sie eine Datei *Aufgabe_07_1b.cpp* an, implementieren und kommentieren Sie Ihren Automaten in C++. Orientieren Sie sich dabei am Beispiel der Vorlesung (siehe Internetseite der Vorlesung). Die Eingabe von zu testenden „Frosch"-Strings können Sie wie folgt gestalten:

```
1  ...
2  int main() {
3    int const size = 256;
4    char s[size];
5    cout << "\"Frosch\"-String eingeben: ";
6    cin >> s;
7    // Test auf Zulaessigkeit
8    bool ok = Akzeptor(s);
9    cout << endl;
10   cout << "\"Frosch\"-String ist ";
11   if (ok)
12     cout << "korrekt!";
13   else
14     cout << "nicht korrekt!";
15   cout << endl;
16   return 0;
17 }
```

c) Testen Sie Ihre Implementierung anhand der oben angegebenen Beispiele für „gültige“ und „ungültige“ „Frosch“-Strings und kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der Datei *Aufgabe_07_1b.cpp* (1 Punkt).

d) In der Vorlesung wurde die Übergabe von Parametern per Kommandozeile besprochen. Kopieren Sie die Datei *Aufgabe_07_1b.cpp* in die Datei *Aufgabe_07_1d.cpp* und passen Sie die in Aufgabenteil (b) erstellte Implementierung so an, dass der zu testende „Frosch“-Strings als Parameter an das Programm übergeben wird (also z.B. `D:\>Aufgabe_07_1c quack`). Vergessen Sie auch hier nicht, Ihren Quelltext zu kommentieren (1 Punkt)!

(Optionale)Aufgabe 2: Grammatik (3 Punkte)

Hinweis: Bei dieser Aufgabe handelt es sich um eine leicht modifizierte Prüfungsaufgabe aus dem Jahr 2007.
Gegeben sei die folgende Grammatik $G = (N, T, S, P)$, wobei

- $N = \{X, Y, Z\}$,
- $T = \{(,), \%, \#, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$,
- $S = X$ gilt und
- P aus

(P1) $X \rightarrow Y$
(P2) $X \rightarrow X\#Y$
(P3) $Y \rightarrow Z$
(P4) $Y \rightarrow Y\%Z$
(P5) $Z \rightarrow 0|1|2|3|4|5|6|7|8|9$
(P6) $Z \rightarrow (X)$

besteht.

Die Produktionen sind zusätzlich mit (P1) bis (P6) gekennzeichnet, damit sie (bei der Ableitung) identifiziert werden können.

a) Sind die folgenden Wörter aus der Grammatik ableitbar? Legen Sie eine Ergebnisdatei *Aufgabe_07_2a.txt* an. Geben Sie jeweils eine Ableitung an, wenn das Wort aus der Grammatik ableitbar ist, oder begründen Sie, warum das Wort nicht ableitbar ist.

   Beispiel: Für die Ableitung von 3 schreiben Sie z.B.

$$X \xrightarrow{(P1)} Y \xrightarrow{(P3)} Z \xrightarrow{(P5)} 3$$

   oder geben einen entsprechenden Ableitungsbaum an.

   (a) Das abzuleitende Wort ist: 3%(5%2) (0,6 Punkte)

   (b) Das abzuleitende Wort ist: ((1%5#6%9))#0) (0,6 Punkte)

   (c) Das abzuleitende Wort ist: (1#2)%(6) (0,6 Punkte)

b) Legen sie eine Ergebnisdatei *Aufgabe_07_2b.txt* an. Erweitern Sie die Grammatik aus Teilaufgabe (a) so, dass auch Ausdrücke mit dem Zeichen '*' ableitbar sind, also auch insbesondere Wörter der Form:

$$4\#(8 * 1)(0,6 Punkte)$$

   Geben Sie *alle* notwendigen Änderungen der Grammatik-Bestandteile an.

c) Legen Sie eine Ergebnisdatei *Aufgabe_07_2c.txt* an. Geben Sie eine Ableitung für das oben angegebene Wort $4\#(8 * 1)$ an. Beschreiben Sie die Ableitung in derselben Form wie in Aufgabenteil (a)(0,6 Punkte).