

Grundlagen der Programmiersprachen und Compilerbau

Wintersemester 2007/2008

2. Übungsblatt

28. November 2007

Aufgabe 2.1

Aus WIKIPEDIA:

Die seit der Einführung der Version 4 des Internet Protocols überwiegend verwendeten IPv4-Adressen bestehen aus 32 Bits, also 4 Oktetts (Bytes). Damit sind 2^{32} , also 4.294.967.296 Adressen darstellbar. In der *dotted decimal notation* werden die 4 Oktetts als vier durch Punkte voneinander getrennte ganze Zahlen in Dezimaldarstellung im Bereich von 0 bis (einschließlich) 255 geschrieben, Beispiel: 130.94.122.195.

Konstruieren Sie einen deterministischen endlichen Automaten, der gültige IP-Adressen in der *dotted decimal notation* akzeptiert. Zur Abkürzung können Sie die Zustandsübergänge des Automaten mit Mengen von Ziffern beschriften.

Aufgabe 2.2

Für die Programmiersprache MANGER[©] existiert folgende Flex-Spezifikation des Scanners [LMB92].

```
%%
wenn                                { return WENN;      }
solange                             { return SOLANGE;  }
dann                                 { return DANN;      }
tu                                  { return TU;         }
ende                                 { return ENDE;       }
[a-z][a-z0-9]*                      { return BEZEICHNER; }
[0-9]+                               { return ZAHL;      }
([0-9]+ "." [0-9]*) | ([0-9]* "." [0-9]+) { return REELE_ZAHL; }
([\t\n]+) | (" ; " [a-z0-9 \t]* \n)   { /* nichts */    }
.                                     { fehler();      }
%%
```

Geben Sie für die einzelnen reguläre Ausdrücke einen endlichen deterministischen Automaten an. Kombinieren Sie diese zu einem endlichen deterministischen Gesamt-Automaten. Kodieren Sie den resultierenden Automaten als eine Tabelle. Können Sie die Größe der Tabelle reduzieren? Wenn ja, wie?

Nach welchen Verfahren oder Ansätzen könnte man Scanner noch implementieren? Welches Verfahren halten Sie für besonders effizient?

Aufgabe 2.3

- a) Konstruieren Sie nach dem Verfahren von Thompson einen (nichtdeterministischen) endlichen Automaten für den regulären Ausdruck $((\varepsilon | a)b^*)^*$.
- b) Wandeln Sie diesen Automaten durch die Untermengenkonstruktion nach Rabin-Scott in einen deterministischen Automaten um.
- c) Wieviele Zustände kann ein DEA, der durch Anwendung des Verfahrens von Rabin-Scott auf einen NEA mit n Zuständen entstanden ist, maximal haben? Welches Problem könnte also bei der praktischen Anwendung dieses Verfahrens auftreten? Tritt das Problem tatsächlich auf?

Aufgabe 2.4

Viele Skriptsprachen (z. B. Perl, Python oder Ruby) bieten den sogenannten *Look-Ahead Operator* an und erlauben somit die Formulierung von regulären Ausdrücken, welche den Folgekontext beachten. Zum Beispiel in oben genannten Sprachen würde der Ausdruck `Isaac_(?=Asimov)` nur dann die Teilfolge `Isaac_` erkennen, falls darauf die Zeichenfolge `Asimov` folgt.

Im Drachenbuch [ASU99] in 3.8 auf S. 161 (deutsche Ausgabe) wird eine mögliche Umsetzung des Look-Ahead-Operators vorgestellt (r_1/r_2 -Notation). Entwickeln Sie einen deterministischen endlichen Automaten für den Ausdruck $(a|ab)/ba$. Testen Sie den Automaten mit `aba` als Eingabe.

Literatur

- [ASU99] Alfred V. Aho, Ravi Seith, and Jeffrey D. Ullmann. *Compilerbau*, volume 1. Oldenbourg, 1999.
- [LMB92] John R. Levine, Tony Mason, and Doug Brown. *lex & yacc*. O'Reilly, 1992.