Technische Universität München
Fakultät für Informatik
Prof. Dr. M. Broy

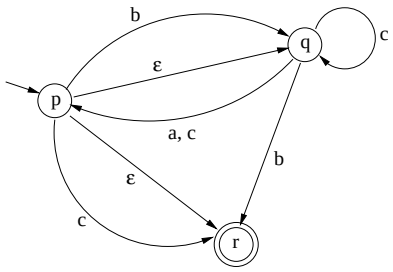
SS 2003
5. Juli 2003

Lösungsvorschläge der Klausur zu Einführung in die Informatik IV

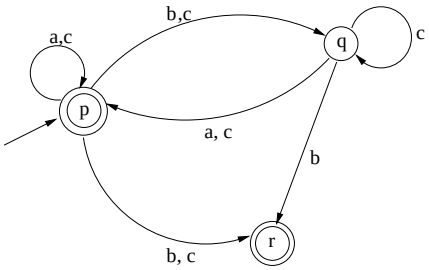
A

Aufgabe 1 Endliche Automaten (Punkte)

(a) Aus der Tabelle ergibt sich zunächst folgender endliche Automat :



Eliminiert man die ε-Übergänge so ergibt sich :



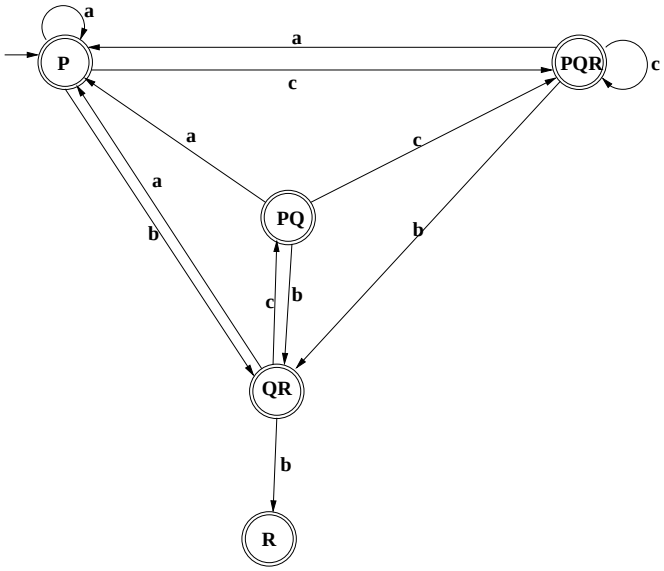
(b) Mit dem Myhill-Verfahren ergibt sich:

	a	b	c
p	{p}	{q, r}	{p, q, r}
{q, r}	{p}	{r}	{p, q}
r	∅	∅	∅
{p, q, r}	{p}	{q, r}	{p, q, r}
{p, q }	{p}	{q, r}	{p, q, r}

Wir erhalten somit den deterministischen Automaten

$$A' = (\{P, R, QR, PQR, PQ\}, \{a, b, c\}, P, \{P, R, QR, PQR, PQ\}, \delta)$$

mit folgender graphischer Darstelllung:



Aufgabe 2 Formale Sprachen

(a) $G = (T, N, \rightarrow, Z)$, wobei $T = \{a, b, \neg, \vee, (,)\}$, $N = \{Z\}$ und $\rightarrow =$
$\{a \rightarrow Z,$
$b \rightarrow Z,$
$\neg Z \rightarrow Z,$
$(Z \vee Z) \rightarrow Z\}$

(b) Die Sprache ist entscheidbar, da die Grammatik kontextfrei ist, d.h. vom Typ Chomsky-2.

Aufgabe 3 Primitiv rekursive Funktionen

$f(0) = 0 \qquad = g()$
$f(succ(n)) = succ(n) + f(n) \qquad = h(n, f(n))$

also ist $f = pr(g,h)$ mit
$g = zero^{(0)}$ und
$h = add \circ [succ \circ \pi_1^2, \pi_2^2]$

Aufgabe 4 Lösung: Zusammenhangskomponente

```
public class Graph {

    // Part a)
    static boolean[][] edges = new boolean[5][5];

    public Graph() {
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                edges[i][j] = false;
            }
```

```
    }
    edges[0][1] = true; edges[1][0] = true;
    edges[1][2] = true; edges[2][1] = true;
    edges[3][4] = true; edges[4][3] = true;
}

// Part b)
public void transitiveClosure() {
    for (int k = 0; k < edges.length; k++) {
        for (int i = 0; i < edges.length; i++) {
            for (int j = 0; j < edges.length; j++) {
                if (edges[i][k] && edges[k][j]) {
                    edges[i][j] = true;
                    edges[j][i] = true;
                }
            }
        }
    }
}

// Part c)
public boolean[] getComponents() {
    transitiveClosure();
    boolean[] result = new boolean[edges.length];
    boolean[] visited = new boolean[edges.length];
    for (int i = 0; i < edges.length; i++) {
        for (int j = 0; j < edges.length; j++) {
            if (edges[i][j] && !visited[j]) {
                result[i] = true;
                visited[j] = true;
            }
        }
    }
    return result;
}

// For testing only
public static void main(String[] args) {
    Graph g = new Graph();
    g.transitiveClosure();
    for (int i = 0; i < edges.length; i++) {
        for (int j = 0; j < edges.length; j++) {
            System.out.print("\t" + edges[i][j]);
        }
        System.out.print("\n");
    }
    boolean[] c = g.getComponents();
    for (int k = 0; k < edges.length; k++) {
        if (c[k])
            System.out.println("Component: " + k);
    }
```

```
    }
}
```