Technische Universität München
Fakultät für Informatik
Dr. M. Schneider, Dr. M. Pizka,
Dr. A. Vilbig, Dr. R. Letz,
G. Bauer, B. Zhang

SS 2003
Lösungsvorschläge zu Blatt 4
9. Mai 2003

Übungen zur Vorlesung Einführung in die Informatik IV

Aufgabe 14 Chomsky-Hierarchie

(a) Wir zeigen: Die Sprache $L = \{a^n b^m c^{n+m} : n \geq 1\}$ ist nicht regulär, aber kontextfrei.

Wir nehmen an, L ist regulär.

Dann gibt es nach dem Pumping-Lemma für reguläre Sprachen eine Konstante $n \in \mathbb{N}$ so, dass sich alle Wörter $z \in L$ mit $|z| \geq n$ zerlegen lassen in $z = uvw$ so, dass gilt:

$$\begin{array}{ll} (i) & |v| \geq 1 \\ (ii) & |uv| \leq n \\ (iii) & uv^iw \in L, \quad \text{für alle } i \geq 1 \end{array}$$

Wir betrachten eine solche Zerlegung des Wortes

$$z = \underbrace{a \cdots a}_{n\text{-mal}}\underbrace{c \cdots c}_{n\text{-mal}}.$$

Wegen (i) und (ii) gilt: $1 \leq |v| \leq n$.

Also ist v ist von der Form $\underbrace{a \cdots a}_{k\text{-mal}}$ mit $1 \leq k \leq n$.

Dann ist $uv^2wv = \underbrace{a \cdots a}_{n+k\text{-mal}} \underbrace{c \cdots c}_{n\text{-mal}} \notin L$, im Widerspruch zu (iii).

Also ist L keine reguläre Sprache.

(b) **(H)** Wir zeigen: L ist eine kontextfreie Sprache.

Die folgende Grammatik akzeptiert genau die Sprache L:

$$G = (T, N, \rightarrow, Z)$$

wobei

$$\begin{array}{rcl} T & = & \{a, b, c\} \\ N & = & \{S, Z\} \\ \rightarrow & = & \{bc \rightarrow S, \\ & & bSc \rightarrow S, \\ & & bSc \rightarrow Z \\ & & aZc \rightarrow Z\} \end{array}$$

(c) Wir zeigen: Die Sprache $L = \{a^n b^n c^n : n \geq 1\}$ ist nicht kontextfrei, aber kontextsensitiv.

Wir nehmen an, L ist kontextfrei.

Dann gibt es nach dem Pumping-Lemma für kontextfreie Sprachen eine Konstante $n \in \mathbb{N}$ so, dass sich alle Wörter $z \in L$ mit $|z| \geq n$ zerlegen lassen in $z = uvwxy$ so, dass gilt:

$$\begin{array}{rll} (i) & |vx| \geq 1 & \\ (ii) & |vwx| \leq n & \\ (iii) & uv^iwx^iy \in L, & \text{für alle } i \geq 1 \end{array}$$

Wir betrachten eine solche Zerlegung des Wortes

$$z = \underbrace{a \cdots a}_{n\text{-mal}}\underbrace{b \cdots b}_{n\text{-mal}}\underbrace{c \cdots c}_{n\text{-mal}}.$$

Wegen (ii) ist vwx ist entweder von der Form $\underbrace{a \cdots a}_{k\text{-mal}}\underbrace{b \cdots b}_{l\text{-mal}}$ oder $\underbrace{b \cdots b}_{k\text{-mal}}\underbrace{c \cdots c}_{l\text{-mal}}$ wobei jeweils $1 \leq k + l \leq n$.

Dann ist in uv^2wx^2y entweder (im ersten Fall) die Anzahl der c's kleiner als die der a's und b's, oder (im zweiten Fall) die Anzahl der a's kleiner als die der b's und c's.

Also ist $uv^2wx^2y \notin L$.

Also ist L keine kontextfreie Sprache.

(d) **(H)** Wir zeigen: L ist eine kontextsensitive Sprache.

Die folgende (wortlängenmonotome) Grammatik akzeptiert genau die Sprache L:

$$G = (T, N, \rightarrow, Z)$$

wobei

$$\begin{array}{rcl} T & = & \{a, b, c\} \\ N & = & \{S, A, Z\} \\ \rightarrow & = & \{bc \rightarrow S, \\ & & bSc \rightarrow SA, \\ & & Ac \rightarrow cA \\ & & aS \rightarrow Z \\ & & aZA \rightarrow Z\} \end{array}$$

Die folgende Grammatik akzeptiert ebenfalls genau die Sprache L:

$$G = (T, N, \rightarrow, Z)$$

wobei

$$\begin{array}{rcl} T & = & \{a, b, c\} \\ N & = & \{A, B, Z\} \\ \rightarrow & = & \{aa \rightarrow aB, \\ & & aaA \rightarrow aB, \end{array}$$

$$Bb \to bB$$
$$Bbcc \to Ac$$
$$bA \to Ab$$
$$aAbc \to Z$$
$$abc \to Z\}$$

Sie ist zwar nicht kontextsensitiv (wegen der Regel $Ab \to bA$), kann aber leicht in eine kontextsensitive Grammatik umgewandelt werden, indem die Regel $Ab \to bA$ ersetzt wird durch die Regeln $Ab \to A'b$, $A'b \to A'A$ und $A'A \to bA$.

Aufgabe 15 Strukturelle Äquivalenz von Ableitungen

(a) Linksnormalform:

$$a\underline{aa}aa \Longrightarrow a\underline{A}aa \Longrightarrow aE\underline{aa} \Longrightarrow a\underline{EA} \Longrightarrow a\underline{E} \Longrightarrow Z$$

Rechtsnormalform:

$$aaa\underline{aa} \Longrightarrow aaa\underline{A} \Longrightarrow a\underline{aa}E \Longrightarrow a\underline{AE} \Longrightarrow a\underline{E} \Longrightarrow Z$$

(b) Die (sequentiellen)Ableitungen des Wortes *abab* lauten:

$$\underline{ab}ab \Longrightarrow Z\underline{ab} \Longrightarrow \underline{ZZ} \Longrightarrow Z \quad (1)$$
$$a\underline{ba}b \Longrightarrow \underline{aZb} \Longrightarrow Z \quad (2)$$
$$ab\underline{ab} \Longrightarrow \underline{ab}Z \Longrightarrow \underline{ZZ} \Longrightarrow Z \quad (3)$$

Die Ableitungen (1) und (3) sind strukturell äquivalent, sie sind aber nicht strukturell äquivalent zu (2).

Aufgabe 16 Kellerautomaten

(a) Die Lösungsidee ist folgende: Ein Eingabewort wird von links nach rechts gelesen. Solange das Zeichen \$ noch nicht gelesen wurde, verbleibt der Automat im Zustand z_0 und füllt den Keller geeignet auf. Wird das Zeichen \$ gelesen, so geht der Automat in den Zustand z_1 über und löscht abhängig vom gerade gelesenen Zeichen den Kellerinhalt.

Der gesuchte Kellerautomat lautet dann:

$$M = (\{z_0, z_1, z_2, \{a, b, \$\}, \{\#, A, B\}, \delta, z_0, \{z_2\})$$

Die Übergangsfunktion lautet dabei folgendermaßen:

$\delta(z_0, a, \#) = (z_0, A\#)$, $\delta(z_0, a, A) = (z_0, AA)$, $\delta(z_0, a, B) = (z_0, AB)$,
$\delta(z_0, b, \#) = (z_0, B\#)$, $\delta(z_0, b, A) = (z_0, BA)$, $\delta(z_0, B, B) = (z_0, BB)$,
$\delta(z_0, \$, \#) = (z_1, \#)$, $\delta(z_0, \$, A) = (z_1, A)$, $\delta(z_0, \$, B) = (z_1, B)$,
$\delta(z_1, a, A) = (z_1, \varepsilon)$, $\delta(z_1, b, B) = (z_1, \varepsilon)$, $\delta(z_1, \varepsilon, \#) = (z_2, \varepsilon)$

(b) Um die Ableitung des Wortes `ba$ab` zu beschreiben, verwenden wir zur Darstellung des momentanen Automatenzustandes den Begriff der Konfiguration eines Automaten:

Eine *Konfiguration* eines Kellerautomaten ist gegeben durch ein Tripel $k \in (S \times T^* \times K^*)$. Eine Konfiguration beschreibt die Momentaufnahme eines Kellerautomaten: Seinen Zustand, das noch zu bearbeitende Wort und den Inhalt des Kellers. Auf der Menge der Konfigurationen ist die Relation $\vdash$ definiert, wobei $k \vdash k'$ genau dann gilt, wenn k' aus k durch Anwendung der δ-Funktion hervorgeht.

Dann gilt folgende Ableitung:

$$(z_0, ba\$ab, \#) \vdash (z_0, a\$ab, B\#) \vdash (z_0, \$ab, AB\#) \vdash \\ (z_1, ab, AB\#) \vdash (z_1, b, B\#) \vdash (z_1, \varepsilon, \#) \vdash (z_2, \varepsilon, \varepsilon)$$

Aufgabe 17 DEA, Minimierung

(a) Minimierungsalgorithmus

Sei $A = \langle S, T, s_0, S_z, \delta \rangle$ ein ε-freier deterministischer endlicher Automat. Die Minimierung erfolgt in zwei Schritten:

- Beseitigung aller vom Anfangszustand nicht erreichbaren Zustände
- Zusammenfassung äquivalenter Zustände

Der erste Schritt lässt sich durch einen üblichen Erreichbarkeitsalgorithmus in Graphen realisieren, deshalb betrachten wir lediglich den zweiten Schritt.

Wir bezeichnen zwei Zustände s_1, s_2 als *k-äquivalent*, wenn für alle Wörter w der Länge $\leq k$ folgendes gilt: es existiert ein mit den Zeichen aus w markierter Pfad von s_1 zu einem Endzustand genau dann, wenn ein solcher Pfad von s_2 zu einem Endzustand existiert. Zwei Zustände heissen *äquivalent*, wenn sie für alle $k \geq 0$ k-äquivalent sind.

Offenbar gilt: s_1 und s_2 sind nur dann $k+l$-äquivalent, wenn sie k-äquivalent sind. Damit können wir durch Inkrementierung von k Schritt für Schritt alle nicht äquivalenten Knotenpaare beseitigen. Wird für eine Stufe k kein Paar beseitigt, muss k nicht weiter inkrementiert werden. Als Datenstruktur benutzen wir eine 2-dimensionale Boolesche Matrix M mit Werten aus {t,f}, die für das jeweils aktuelle k die entsprechenden Paare beseitigt. Für $k = 0$ werden alle Matrixwerte $M(s_i, s_j)$ mit t markiert, für die entweder s_i, s_j beide Endzustände oder beide keine Endzustände sind (wegen der Symmetrie reicht eine Hälfte der Matrix).

Initialisierung der Matrix (Zustände seien $s_0, s_1, \ldots$)

```
forall 0 ≤ i < |S|
   forall i < j < |S|
      if s_i ∈ S_Z then
         if s_j ∈ S_Z then M(s_i,s_j) := t  else M(s_i,s_j) := f
         else if s_j ∈ S_Z then M(s_i,s_j) := f  else M(s_i,s_j) := t  fi
         fi
      fi
   end
end
```

Berechnung der Äquivalenzpaare

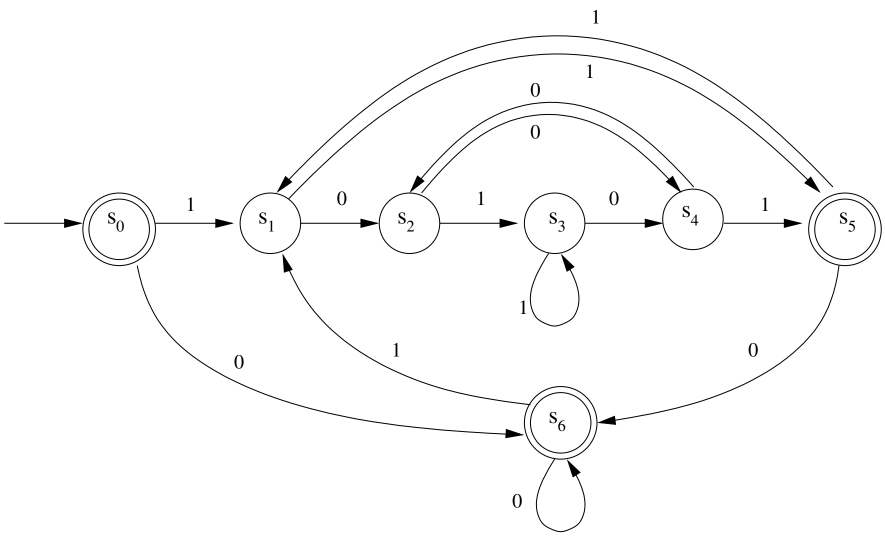
```
while t
   pair_eliminated := f;
```

forall $0 \leq i < |S|$
forall $i < j < |S|$
if $M(s_i, s_j) = \mathrm{t}$ then
forall $a \in T$
if $M(\delta(s_i, a), \delta(s_j, a)) = \mathrm{f}$ then
$M(s_i, s_j) := \mathrm{f}$; pair_eliminated := t
fi
end
if
end
end;
if not pair_eliminated then exit fi
end

Eigentliche Minimierung des Automaten

forall $0 \leq i < |S|$
forall $i < j < |S|$
if $M(s_i, s_j) = \mathrm{t}$ then
forall in s_i gehende Kanten k: leite k nach s_j um (falls k nicht doppelt);
entferne den Knoten s_i
fi
end
end

(b) Anwendung des Algorithmus auf den folgenden Automaten:



Entwicklung der Äquivalenzmatrix für $k = 0, 1$ (nur t eingetragen):

$k = 0$:

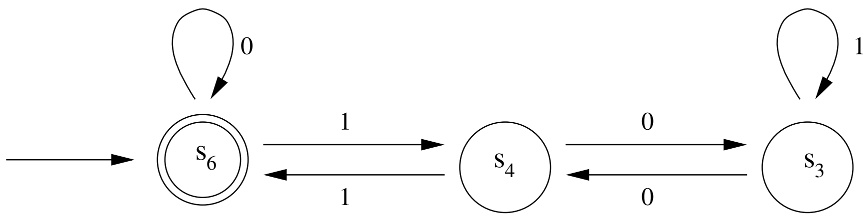
$i \setminus j$	s_0	s_1	s_2	s_3	s_4	s_5	s_6
s_0	t					t	t
s_1		t	t	t	t		
s_2			t	t	t		
s_3				t	t		
s_4					t		
s_5						t	t
s_6							t

$k = 1$:

$i \setminus j$	s_0	s_1	s_2	s_3	s_4	s_5	s_6
s_0	t					t	t
s_1		t			t		
s_2			t	t			
s_3				t			
s_4					t		
s_5						t	t
s_6							t

Für $k = 2$ ergibt sich keine Änderung mehr (d.h. pair_eliminated = f) und das Verfahren terminiert mit 3 Äquivalenzklassen.

Die eigentliche Minimierung ergibt den folgenden Automaten, der isomorph zur angegeben Lösung von 11 (c) ist:



Aufgabe 18 (P) Minimierung eines endlichen Automaten

Die vorgestellte Lösung folgt in etwa dem Algorithmus aus Aufgabe 17. Die erforderlichen Ergänzungen in der Repräsentation eines Automaten werden als Vorlage im WWW zur Verfügung gestellt.

Teilaufgabe a)

```
// Returns a boolean matrix with marked equivalent state pairs
private boolean[][] calculateEquivalenceMatrix() {
    boolean[][] marked = new boolean[numStates][numStates];
    // Use lower triangle of matrix as equivalence is reflexive and symmetric
    for (int i = 0; i < numStates; i++)
        for (int j = 0; j < i; j++)
            marked[i][j] = false;
    // Init: set homogeneous terminal/non-terminal state pairs as equivalent
    for (int i = 0; i < numStates; i++) {
        for (int j = 0; j < i; j++) {
            State si = (State) stateList.get(i);
            State sj = (State) stateList.get(j);
            if ((si.isTerminal() && sj.isTerminal()) ||
                (!si.isTerminal() && !sj.isTerminal()))
                marked[i][j] = true;
        }
    }
    // Elongation: check labels and state pairs for possible transitions
    boolean isStable = false;
    while (!isStable) {
        isStable = true;
        for (int i = 0; i < numStates; i++) {
            for (int j = 0; j < i; j++) {
                if (marked[i][j]) {
                    Iterator labelIt = labels.iterator();
                    while (labelIt.hasNext()) {
                        Label l = (Label) labelIt.next();
                        Set rm = automaton[i][l.getId()];
                        Set rn = automaton[j][l.getId()];
                        // Automaton deterministic & total -> well-defined result
```

```
                        State sm = (State) rm.iterator().next();
                        State sn = (State) rn.iterator().next();
                        int m = Math.max(sm.getId(), sn.getId());
                        int n = Math.min(sm.getId(), sn.getId());
                        // Check for transition into non-equivalent state pair,
                        // excluding reflexive pairs (which are always equivalent)
                        if (!marked[m][n] && m != n) {
                            marked[i][j] = false;
                            isStable = false;
                        }
                    }
                }
            }
        }
    }
    return marked;
}
```

Teilaufgabe b)

```
// Minimize automaton from given matrix of equivalent state pairs
private void fromEquivalenceMatrix(boolean[][] matrix) {
    System.out.println(matrixToString(matrix));
    Set removed = new HashSet();
    for (int i = 0; i < numStates; i++) {
        for (int j = 0; j < i; j++) {
            if (matrix[i][j]) {
                // Marked state pairs are equivalent
                State si = (State) stateList.get(i);
                State sj = (State) stateList.get(j);
                // Both states still relevant?
                if (!removed.contains(si) && !removed.contains(sj)) {
                    System.out.println("Merging " + si + " and " + sj +
                                       " [" + i + "," + j + "]");
                    // Redirect incoming transitions of si to sj
                    Iterator incomingIt = si.getIncomingTransitions().iterator();
                    while (incomingIt.hasNext()) {
                        String from;
                        Transition t = (Transition) incomingIt.next();
                        // Check for loop transitions
                        if (t.getOrigin() != t.getTarget())
                            from = t.getOrigin().getName();
                        else
                            from = sj.getName();
                        System.out.println("Redirecting " + t + " from " + from +
                                           " to " + sj + " (instead of " + si + ")");
                        addTransition(from, sj.getName(), t.getLabel().getName());
                    }
                    // Transfer start property if necessary
                    if (si.isStart())
                        sj.setStart(true);
```

```
                    // Schedule processed state for later removal
                    System.out.println("Removing " + si);
                    removed.add(si);
                }
            }
        }
    }
    // Delete processed states
    Iterator remIt = removed.iterator();
    while (remIt.hasNext()) {
        State s = (State) remIt.next();
        removeState(s.getName());
    }
}
```

Teilaufgabe c)

```
public void minimize() {
    fromEquivalenceMatrix(calculateEquivalenceMatrix());
    init();
}
```