Universität Karlsruhe (TH)

Institut für Innovatives Rechnen und Programmstrukturen (IPD)

Übersetzerbau WS 2003/04 http://www.info.uni-karlsruhe.de/

Dozent: Prof. Dr.rer.nat. G. Goos goos@ipd.info.uni-karlsruhe.de

Übungsleiter: Rubino Geiß rubino@ipd.info.uni-karlsruhe.de

Übungsblatt 2 | Ausgabe: 28.10.2003 | Besprechung: 06.11.2003

Aufgabe 1: *Reguläre Ausdrücke – Endliche Automaten*

1.1 *Reguläre Ausdrücke*

Geben Sie einen regulären Ausdruck für Bezeichner an! Bezeichner beginnen mit einem Buchstaben. Es folgt eine beliebige Zeichenreihe aus Buchstaben, Ziffern und "_", wobei "_" nicht als abschließende Zeichen und keine zwei "_" hintereinander vorkommen dürfen.

1.2 *Konstruktion*

Konstruieren Sie einen endlichen Automaten der die obige Sprache akzeptiert. Gehen Sie dabei nach dem Konstruktionsverfahren in aus der Vorlesung vor.

1.3 *Minimaler endlicher Automat*

Reduzieren Sie den konstruierten endlichen Automaten!

Aufgabe 2: *Reguläre Ausdrücke*

E bezeichne die leere Sprache, *eps* die Sprache, die nur die leere Zeichenkette ϵ enthält. $X + Y$ bezeichne die Vereinigung der durch X und Y bezeichneten Sprachen, XY die Sprache, die alle Konkatenationen von Zeichenketten aus X und Y enthält. Was ist dann

1. EX
2. $epsX$
3. $E + X$
4. $eps + X$

für eine beliebige Menge X von Zeichenketten?

Aufgabe 3: *Korrespondenz reguläre Grammatiken - endliche Automaten*

Geben sei folgende reguläre Grammatik: $G = (\{X, Y, Z\}, \{1, 2\}, R, X)$ mit

$$R = \{(X \to 1X), (X \to Y), (Y \to 2Y), (Y \to 2Z), (Z \to 2Z)(Z \to 2)\}$$

3.1 *Konstruktion*

Konstruieren Sie den entsprechenden endlichen Automaten, der genau die durch G definierte Sprache akzeptiert.

3.2 *NEA / DEA*

Welche Produktionen haben Spontanübergänge im Automaten erzeugt? Eliminieren Sie diese Spontanübergänge im Automaten. Überführen Sie den so erhaltenen Automaten in einen äquivalenten deterministischen Automaten.

Aufgabe 4: *Reguläre Grammatiken – Endliche Automaten*

Geben sei folgende Grammatik $G = (\{X, Y\}, \{zi, .\}, R, X)$ mit:

$$R = \{(X \to Y.Y), (X \to .Y), (Y \to zi), (Y \to ziY)\}$$

4.1 *Reduzieren der Grammatikklasse*

Transformieren Sie die Grammatik so, daß nur noch Regeln der Formen $(A \to B)$, $(A \to aB)$, $(A \to a)$ und $(A \to \epsilon)$ vorkommen (A und B sind nichtterminale Symbole, a ist ein terminales Symbol, $A = B$ ist möglich)!

Aufgabe 5: *Haschen*

Unter welchen Bedingungen ist der Eintrag eines Symbols in die Symboltabelle fester Größe

1. im schlechtesten Fall,
2. im erwarteten Fall,

$\mathcal{O}(1)$. Sind das realistische Annahmen für Programmiersprachen?

Wie groß ist der erwartete Aufwand, wenn die Tabelle jeweils verdoppelt wird, sobald sie zu mehr als z.B. 70% gefüllt ist? (Wir nehmen an, daß jede der gewählten Hasch-Funktionen die Einträge gleichverteilt.)

Aufgabe 6: *Generatoren*

Gesucht ist ein Symbolentschlüsseler, der Bezeichner (siehe Aufgabe 2.1) erkennt und in eine Symboltabelle einträgt. Die Sprache läßt neben Bezeichnern lediglich Leerzeichen zu.

6.1 *Praxis Test*

Wie sieht die Spezifikation für einen Generator Ihrer Wahl aus?

6.2 *Schnittstellen*

Wie sieht die Schnittstelle des generierten Programms aus?

Aufgabe 7: *$LL(k)$ und $SLL(k)$*

Eine kontextfreie Grammatik $G = (T, N, P, Z)$ heißt *SLL(k)*, $k > 0$, falls für beliebige Ableitungen

$$Z \Rightarrow^L \mu A \chi \Rightarrow \mu\nu\chi \Rightarrow^* \mu\gamma \qquad \mu, \gamma \in T^*, \nu, \chi \in V^*, A \in N$$
$$Z \Rightarrow^L \mu' A \chi' \Rightarrow \mu'\omega\chi' \Rightarrow^* \mu'\gamma' \qquad \mu', \gamma' \in T^*, \omega, \chi' \in V^*$$

aus $(k : \gamma = k : \gamma')$ $\nu = \omega$ folgt.

Zeigen Sie:

7.1

Eine kontextfreie Grammatik $G = (T, N, P, Z)$ ist SLL(1) gdw. für alle Produktionen $A \to l_1$, $A \to l_2$ gilt:

$$Anf(l_1 Folge(A)) \cap Anf(l_2 Folge(A)) = \emptyset.$$

7.2

Wie kann man $Folge_k(A)$ und $Anf_k(A)$ effizient berechnen?

7.3

Jede SLL(k)-Grammatik ist auch LL(k).

7.4

Eine kontextfreie Grammatik $G = (T, N, P, Z)$ ist LL(1) genau dann, wenn sie SLL(1) ist. Diese Aussage gilt nicht für beliebige k.

7.5

Für welches k sind die folgenden Grammatiken LL(k)? Sind sie auch SLL(k)?

(a)
$Z \to S$
$S \to aS$
$S \to a$

(b)
$Z \to S$
$S \to aA$
$A \to S$
$A \to \varepsilon$

(c)
$Z \to C \mid D$
$C \to aC \mid b$
$D \to aD \mid c$

(d)
$Z \to S$
$S \to Ax \mid By \mid dAy$
$A \to C \mid z$
$B \to C$
$C \to c$

7.6

Grammatiken sind nicht alle SLL(2) (Gegenbeispiel).

7.7

Die Ausdrucksgrammatik (wie in der Vorlesung gegeben) ist weder $SLL(k)$ noch $LL(k)$ für beliebige k?

Aufgabe 8: *LL(1)-Automaten*

Gegeben sei die folgende Grammatik der booleschen Ausdrücke:

(1) $Z \to F$
(2) $F \to D \mid N$
(3) $D \to D$ **or** $K \mid K$
(4) $N \to$ **not** F
(5) $K \to K$ **and** $A \mid A$
(6) $A \to (F) \mid$ **id**

8.1 *Vereinfachen der Grammatik*

Ist diese Grammatik LL(1)? Falls nein, so konstruieren Sie eine äquivalente LL(1)-Grammatik.

8.2 *LL-Kellerautomat*

Konstruieren Sie den dazugehörigen Kellerautomaten.

8.3 *Direkte Implementierung*

Entwickeln Sie einen LL(1)-Zerteiler nach der Methode des rekursiven Abstiegs.