Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr

SS 2002
Übungsblatt 5
17. Mai 2002

Einführung in die Informatik IV

Abgabetermin: Dienstag, 28. Mai 2002, Briefkasten bei S0314

Aufgabe 1

Wir erinnern an die Definition eines 2DEA (Übungsblatt 4, Aufgabe 1). Das Eingabeband besteht aus Feldern $F_0F_1 \dots F_{k+1}$ (k ist die Länge der Eingabe w) und und ist mit $\$_l w \$_r$ beschrieben. Wir betrachten Kreuzungsfolgen bei 2DEAs, die wie folgt definiert sind: Zu jedem Zwischenraum zwischen zwei Feldern F_iF_{i+1} des Eingabebandes definieren wir die dazugehörende Kreuzungsfolge c_i als die Folge der Zustände q', die entstehen, wenn der Lesekopf über Feld F_i steht, $\delta(q, T_i) = (q', R)$ gilt, wobei T_i das Zeichen ist, das in dem Feld F_i enthalten ist, oder entsprechend für den Fall, dass der Lesekopf über Feld F_{i+1} steht, $\delta(q, T_{i+1}) = (q', L)$ gilt, wobei T_{i+1} das im Feld F_{i+1} enthaltene Zeichen ist.
Zeigen Sie, dass alle Kreuzungsfolgen endlich sind, falls der 2DEA terminiert.
Begründen Sie, warum $c_i \leq 2|Q| - 1$ gilt.

Aufgabe 2

Zeigen Sie: Jede Sprache die von einem 2DEA erkannt wird, ist regulär (Hinweis: Fassen Sie die Kreuzungsfolgen als Zustände eines NEA auf). Gilt auch die Umkehrung?

Aufgabe 3

Sei L eine reguläre Sprache. Welche der folgenden Mengen sind regulär:
(a) $\{a_1a_3a_5 \dots a_{2n-1} | a_1a_2a_3 \dots a_{2n} \in L, a_i \in \Sigma\}$
(b) $\{a_2a_1a_4a_3a_6a_5 \dots a_{2n}a_{2n+1} | a_1a_2 \dots a_{2n+1} \in L, a_i \in \Sigma\}$
(c) $CYCLE(L) = \{uv | vu \in L$ für Zeichenketten $u, v \in \Sigma^*\}$
(d) $MIN(L) = \{x \in L|$ kein echtes Präfix von x ist in L $\}$

Aufgabe 4

In dieser Aufgabe soll mit Hilfe des Tools `bison` (oder `yacc`) ein Parser programmiert werden, der bei Eingabe eines Wortes entscheidet, ob dieses Wort in einer vorgegebenen Sprache L ist oder nicht.
(Alle wichtigen Informationen zu `bison` finden Sie auf der Homepage der Vorlesung!)
Wir betrachten das Alphabet $\Sigma = \{\texttt{X,0,1,...,9,.,+,-.*,/,\^{},<,>,=,C,(,),BLANK}\}$ und definieren die Sprache L wie folgt rekursiv:

- Eine **Variable** (Token VAR) ist ein Wort der Form `Xn`, wobei `n` die Dezimaldarstellung einer natürlichen Zahl ist, (z.B. `X0,X1,X1024,...`).
  Keine weiteren Wörter sind Variablen.

- Eine **Zahl** (Token NUM) ist ein Wort, das eine ganze Zahl (zur Basis 10) oder einen (endlichen) Dezimalbruch darstellt,
  (z.B. `12.345, 0.113, -1.0, -20,...`).
  Keine weiteren Wörter sind Zahlen.

- Ein **Term** (oder Ausdruck, oder Expression) ist wie folgt rekursiv definiert:

  - Jede Variable und jede Zahl ist ein Term.
  - Sind T_1 und T_2 zwei Terme, so sind auch (T_1 `+` T_2) und T_1 `*` T_2 Terme.
  - Keine weiteren Wörter sind Terme.

  Beispiele für Terme: `1+2`, `X1*3`, `(X11+X1)*X2`.
  Keine Terme sind z.B. `((1+2)`, `X11-X`, `1.2X4+X12`.

- Eine **Formel** ist wie folgt rekursiv definiert:

  - Sind T_1, T_2 Terme, so ist T_1 `=<` T_2 eine Formel.
    (T_1 `=<` T_2 entspricht $T_1 \leq T_2$.)
  - Sind φ_1 und φ_2 Formeln, so sind auch
    `C` φ_1, φ_1 `&` φ_2 und (φ_1)
    Formeln.
    (`C` φ entspricht $\neg\ \varphi$, d.h. logische Negation.)
  - Sind φ eine Formel und `V` eine Variable, so sind auch
    `\F V` φ und `\E V` φ
    Formeln.
    ("`\F V`" (bzw. "`\E V`") entspricht dem Quantor "$\forall\ V$" (bzw. "$\exists\ V$").)
  - Keine weiteren Wörter sind Formeln.

  Beispiele für Formeln: `(X1+5)=<(X2+3)`, `2*(X1+3)=<X1`, `\E X1 (X1*X1+X2)=<0`
  Keine Formeln sind z.B. `X1+2`, `\F X1=1`.

- Die Sprache L ist nun die Menge aller Terme und Formeln.

Lesen Sie das Manual `bison.ps` (`bison.pdf`) und ergänzen Sie in der vorprogrammierten bison-Datei `logic.y` den Abschnitt *Grammar rules* um die Regeln für die Nicht-Terminale *formula* und *exp*, d.h. für Formeln und Ausdrücke.

Übersetzen Sie die Quelldatei `logic.y` mit `bison logic.y`. Hierbei generiert `bison` den C-Quellcode `logic.tab.c`. Compilieren Sie den C-Quellcode mit `cc logic.tab.c -o logic` und starten Sie das Binary `logic`.

Geben Sie einen Term oder eine Formel ein. Der Parser `logic` erkennt dann entweder den Term oder die Formel oder gibt eine Fehlermeldung aus.

Senden Sie die Datei `logic.y` per Email an Ihren Tutor unter Angabe von Name, Mat.Nr., und GruppenNr.!