Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

12. Übungsblatt, Sommersemester 2015
Abgabetermin: 14.07.2015

Aufgabe 32

Übersetzen Sie den arithmetischen Ausdruck „`a*-(b+c)`“ in 3-Adress-Code.

Aufgabe 33

Entwickeln Sie für die folgende Teilgrammatik für einfache arithmetische Zuweisungen ein Übersetzungsschema zur Erzeugung von 3-Adress-Code.

$S \rightarrow$ **id** := E
$E \rightarrow E_1$ `+` E_2
$E \rightarrow E_1$ `*` E_2
$E \rightarrow$ `-` E_1
$E \rightarrow$ `(` E_1 `)`
$E \rightarrow$ **id**

Verwenden Sie eine Funktion *newtemp()* zur Generierung temporärer Variablen.

Aufgabe 34

a) Entwickeln Sie ein Übersetzungsschema (analog zur Vorlesung) für die Schleife

doloop → **do** *stmt* **end while** `(` *boolexpr* `)`

b) Übersetzen Sie

do
$a = a + b$
end
while `(` $a < b$ `)`

Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

11. Übungsblatt, Sommersemester 2015
Abgabetermin: 07.07.2015

Aufgabe 30

a) Übersetzen Sie die folgenden Definitionen in C-Maschinen-Code:

```
int i;
struct list {
  int info;
  struct list *next;
} *l, *t;

int ithElement(struct list *l, int i){
  if (i == 1)
    return l->info;
  else
    return ithElement(l->next, i - 1);
}
```

b) Übersetzen Sie das folgende zugehörige Programmstück in C-Maschinen-Code:

```
i = ithElement(l,3);
```

Aufgabe 31

In der Programmiersprache C bewirkt ein `break` einen Sprung an das Ende der nächsten umfassenden Schleife bzw. des nächsten umfassenden `switch`-Statements. Wie kann man die Übersetzungsvorschriften erweitern, damit `break`s an beliebiger Stelle richtig übersetzt werden?

Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

10. Übungsblatt, Sommersemester 2015
Abgabetermin: 30.06.2015

Aufgabe 28

Entwickeln Sie ein syntaxgerichtetes Übersetzungsschema, das arithmetische Ausdrücke konvertiert

a) von Postfix- nach Infix-Notation und

b) von Postfix- nach Präfix-Notation.

Geben Sie jeweils für die Eingaben `9 5 - 2 *` und `9 5 2 * -` attributierte Parse-Bäume an.

Aufgabe 29

Sei $\varphi(a) = 5$, $\varphi(b) = 6$ und $\varphi(c) = 7$. Bestimmen Sie

a) $code\ (a = ((a + b) == c))\ \varphi$

b) $code_R\ (a + (a + (a + b)))\ \varphi$

c) $code_R\ (((a + a) + a) + b)\ \varphi$

d) Wieviele Stackzellen werden bei der Ausführung der im Fall b) und der im Fall c) erzeugten Befehlsfolgen belegt?

Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

9. Übungsblatt, Sommersemester 2015
Abgabetermin: 23.06.2015

Aufgabe 25

Prüfen Sie für die folgenden Grammatiken, ob sie in SLR(1) bzw. in LALR(1) sind.

a)
$$\begin{array}{lcl} S' & \to & S \\ S & \to & S \ (\ S\ ) \mid \varepsilon \end{array}$$

b)
$$\begin{array}{lcl} S & \to & X \\ X & \to & M\ \mathbf{a} \mid \mathbf{b}\ M\ \mathbf{c} \mid \mathbf{d}\ \mathbf{c} \mid \mathbf{b}\ \mathbf{d}\ \mathbf{a} \\ M & \to & \mathbf{d} \end{array}$$

Aufgabe 26

Betrachten Sie die Grammatik $G = (V_N, V_T, P, S)$:

- $V_N = \{S\}$
- $V_T = \{\mathbf{a}, \mathbf{e}, \mathbf{i}\}$
- $P = \{S \to \mathbf{i}S\mathbf{e}S,\ S \to \mathbf{i}S, S \to \mathbf{a}\}$

Erstellen Sie die SLR-Parsertafel. Interpretieren Sie die auftretenden shift/reduce-Konflikte und erläutern Sie, was die möglichen Lösungen bedeuten. Parsen Sie **iiaea**.

Aufgabe 27

Gegeben sei die Grammatik G zur Darstellung reeller Zahlen im Exponentialformat durch

$$\begin{array}{lcl} S & \to & A\ \mathbf{.}\ B\ \mathbf{e}\ A \\ A & \to & V\ B \\ V & \to & + \mid \mathbf{-} \\ B & \to & B\ \mathbf{ziffer} \mid \mathbf{ziffer} \end{array}$$

Es wird angenommen, dass **ziffer** als Ergebnis der lexikalischen Analyse entsteht und in dem Attribut **ziffer**.lexval der ermittelte Ziffernwert zu finden ist. Geben Sie eine syntaxgesteuerte Definition an, die den Zahlenwert von S im Fließkommaformat bestimmt.

Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

8

Compilerbau

8. Übungsblatt, Sommersemester 2015
Abgabetermin: 16.06.2015

Aufgabe 23

Betrachten sie die kontextfreie Grammatik $G = (V_N, V_T, P, S')$ mit $V_N = \{S', S, A, B\}$, $V_T = \{\mathbf{a}, \mathbf{b}\}$ und $P\{S' \to S,\ S \to A\mathbf{a}A\mathbf{b} \mid B\mathbf{b}B\mathbf{a},\ A \to \varepsilon,\ B \to \varepsilon\}$.

a) Gilt $G \in LR(1)$?

b) Gilt $G \in SLR(1)$?

Aufgabe 24

Gegeben sei folgende mehrdeutige Grammatik $G_n = (N, T_n, S, P_n)$ für arithmetische Ausdrücke mit n binären Operatoren $\theta_1, \ldots, \theta_n$: $N = \{S, E\}$, $T_n = \{$ `id`, `(`, `)`, θ_1, ..., θ_n, `$`$\}$, mit den Produktionen P_n

$S \quad \to \quad E$ `$`
$E \quad \to \quad E\theta_1 E \mid E\theta_2 E \mid \ldots \mid E\theta_n E \mid$ `(` E `)` $\mid$ `id`

Es gelte: θ_i hat höhere Priorität als $\theta_j \Leftrightarrow i > j$.

a) Konstruieren Sie den LR(0)-Automaten $dchar_0(G_n)$. Geben Sie die Anzahl der Zustände als Funktion von n an.

b) Stellen Sie die SLR(1)-Action-Tabelle auf. Die sich ergebenden Konflikte werden wie folgt aufgelöst:

> Aktionen mit θ_i haben Vorrang vor Aktionen mit θ_j, wenn $i > j$. Bei gleichem Operator soll vorrangig reduziert werden, d.h. die Operatoren sollen linksassoziativ sein.

c) Parsen Sie die Worte „`id` θ_i `id` θ_i `id$`“ (i beliebig) und „`id` θ_i `id` θ_j `id$`“ für $i < j$ und für $i > j$.

Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

7. Übungsblatt, Sommersemester 2015
Abgabetermin: 09.06.2015

Aufgabe 20

Gegeben sei die folgende Grammatik G.

$$\begin{array}{lcl} S & \to & V\ A\ \mathbf{\$} \\ V & \to & + \mid - \\ A & \to & A + T \mid T \mid V\ T \\ T & \to & \mathbf{id} \mid V\ \mathbf{id} \end{array}$$

Bestimmen Sie die zuverlässigen Präfixe von G, den deterministischen charakteristischen Automaten $dchar_G$ zu G sowie die Action-Tabelle $\mathcal{A}_\mathcal{G}$ und die Goto-Tabelle $\mathcal{G}_\mathcal{G}$. Parsen Sie das Eingabewort $+ + \mathbf{id} + - \mathbf{id}$.

Aufgabe 21

Folgende vier kontextfreie Grammatiken erzeugen die reguläre Sprache zu $\mathbf{a}(\mathbf{;a})^*$ (es sei jeweils $S \to L\mathbf{\$}$ die Startproduktion):

a) $L \to L\mathbf{;a}$, $L \to \mathbf{a}$
b) $L \to \mathbf{a;}L$, $L \to \mathbf{a}$
c) $L \to L\mathbf{;}L$, $L \to \mathbf{a}$
e) $L \to \mathbf{a}T$, $T \to \mathbf{;}L \mid \varepsilon$

Welche der Grammatiken sind $LR(0)$? Stellen Sie für diese die Action- und Goto-Tabellen auf und parsen Sie die Eingaben **a;a;a$** und **a;aa$**.

Aufgabe 22

Gegeben sei die Grammatik G aus Aufgabe 19:

$$\begin{array}{lcl} S' & \to & S\ \mathbf{\$} \\ S & \to & S\ \mathbf{b} \mid \mathbf{b}\ A\ \mathbf{a} \\ A & \to & \mathbf{a}\ S\ \mathbf{c} \mid \mathbf{a}\ S\ \mathbf{b} \mid \mathbf{a} \end{array}$$

Berechnen Sie den deterministischen charakteristischen Automaten zu G. Welche Zustände sind ungeeignet für die $LR(0)$-Analyse?

Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

6. Übungsblatt, Sommersemester 2015
Abgabetermin: 02.06.2015

Aufgabe 17

Gegeben sei die folgende Grammatik G.

$$
\begin{array}{lcl}
S & \rightarrow & \mathbf{u}\ B\ D\ \mathbf{z} \\
B & \rightarrow & B\ \mathbf{v} \mid \mathbf{w} \\
D & \rightarrow & E\ F \\
E & \rightarrow & \varepsilon \mid \mathbf{y} \\
F & \rightarrow & \varepsilon \mid \mathbf{x}
\end{array}
$$

a) Berechnen Sie die Start- und Folgemengen für G. Konstruieren Sie die LL(1)-Parse-Tabelle und zeigen Sie, dass G nicht die LL(1)-Eigenschaft besitzt.

b) Konstruieren Sie aus G eine Grammatik G' mit $L(G) = L(G')$ so, dass G' die LL(1)-Eigenschaft hat. Versuchen Sie dabei, die Unterschiede zwischen G und G' so gering wie möglich zu halten.

Aufgabe 18

Berechnen Sie die Parse-Tabelle $\mathcal{T}_G$ für die Grammatik G gegeben durch

$$
\begin{array}{lcl}
S & \rightarrow & \mathbf{if}\ E\ \mathbf{then}\ S\ S' \mid \mathbf{a} \\
S' & \rightarrow & \mathbf{else}\ S \mid \varepsilon \\
E & \rightarrow & \mathbf{b}
\end{array}
$$

Wie kann die Mehrdeutigkeit in der Parse-Tabelle geschickt aufgelöst werden?

Aufgabe 19

Gegeben sei die erweiterte kontextfreie Grammatik $G = (N, T, S', P)$ mit $N = \{S', S, A\}$, $T = \{\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{\$}\}$ und den Produktionen

$$
\begin{array}{lcl}
S' & \rightarrow & S\ \mathbf{\$} \\
S & \rightarrow & S\ \mathbf{b} \mid \mathbf{b}\ A\ \mathbf{a} \\
A & \rightarrow & \mathbf{a}\ S\ \mathbf{c} \mid \mathbf{a}\ S\ \mathbf{b} \mid \mathbf{a}
\end{array}
$$

Berechnen Sie die zugehörigen Linkskontexte und die Menge der maximalen zuverlässigen Präfixe pre_G.

Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

5. Übungsblatt, Sommersemester 2015
Abgabetermin: 26.05.2015

Aufgabe 13

Ist die Menge der kontextfreien Sprachen unter dem Durchschnitt abgeschlossen? Beweisen Sie Ihre Antwort.

Aufgabe 14

Gegeben sei die folgende Grammatik G.

$$
\begin{array}{lcl}
S' & \to & S\ \textbf{\$} \\
S & \to & \varepsilon \mid X\ S \\
B & \to & \textbf{\textbackslash\ begin \{ WORD \}} \\
E & \to & \textbf{\textbackslash\ end \{ WORD \}} \\
X & \to & B\ S\ E \mid \textbf{\{}\ S\ \textbf{\}} \mid \textbf{WORD} \mid \textbf{begin} \mid \textbf{end} \mid \textbf{\textbackslash\ WORD}
\end{array}
$$

a) Berechnen Sie die First- und Followmengen für G.

b) Konstruieren Sie die LL(1)-Parse-Tabelle.

Aufgabe 15

Gegeben sei die folgende Grammatik für reguläre Ausdrücke.

$$
\begin{array}{lcl}
E & \to & E\ \texttt{+}\ T \mid T \\
T & \to & T\ F \mid F \\
F & \to & \texttt{(}\ E\ \texttt{)} \mid \texttt{id} \mid F\ \texttt{*}
\end{array}
$$

Die Terminalzeichen sind `+`, `(`, `)`, `id` und `*`.

a) Eliminieren Sie mit dem Verfahren aus Abschnitt 2.7.3 die Linksrekursion.

b) Berechnen Sie die Start- und Folgemengen und erstellen Sie damit die Parse-Tabelle.

c) Veranschaulichen Sie für die Eingaben „`( id + id ) *`“ und „`id id ( *`“ die Aktionen des Parsers mittels der Darstellung des Kellerinhaltes und des noch nicht gelesenen Teils des Eingabewortes.

Aufgabe 16

Konstruieren Sie einen Rekursiver-Abstieg-Parser für die Grammatik

$$
\begin{array}{lcl}
S & \to & \texttt{id =}\ E \\
E & \to & T\ Q \\
Q & \to & \texttt{+}\ T\ Q \mid \texttt{-}\ T\ Q \mid \varepsilon \\
T & \to & F\ R \\
R & \to & \texttt{*}\ F\ R \mid \texttt{/}\ F\ R \mid \varepsilon \\
F & \to & \texttt{(}\ E\ \texttt{)} \mid \texttt{id}
\end{array}
$$

Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

4. Übungsblatt, Sommersemester 2015
Abgabetermin: 19.05.2015

Aufgabe 10

Berechnen Sie die Start- und Folgemengen für die folgende Grammatik:

$$
\begin{array}{lcl}
S & \rightarrow & A\ B\ C \\
A & \rightarrow & \texttt{a} \mid C\ \texttt{b} \mid \varepsilon \\
B & \rightarrow & \texttt{c} \mid \texttt{d}\ A \mid \varepsilon \\
C & \rightarrow & \texttt{e} \mid \texttt{f}
\end{array}
$$

Aufgabe 11

Betrachten Sie die kontextfreie Grammatik G:

$$S \rightarrow S\ \texttt{a}\ \texttt{a} \mid S\ \texttt{a}\ \texttt{b} \mid \texttt{a}$$

Konstruieren Sie eine LL(1)-Grammatik G', die äquivalent zu G ist. Beweisen Sie, dass $G' \in \text{LL}(1)$.

Aufgabe 12

Sei $L = \{x \in \{(,)\}^* \mid x \text{ ist ein korrekter Klammerausdruck}\}$. Beweisen Sie, dass L nicht regulär ist.

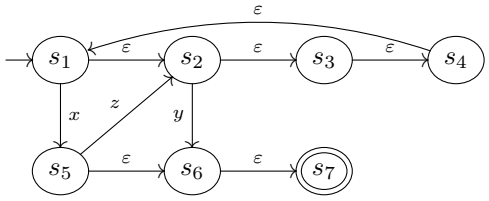
Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

3. Übungsblatt, Sommersemester 2015
Abgabetermin: 12.05.2015

Aufgabe 8

Konstruieren Sie für den folgenden nichtdeterministischen endlichen Automaten den dazugehörigen deterministischen endlichen Automaten.



Aufgabe 9

Ein lineares Optimierungsproblem besteht aus einer zu maximierenden oder minimierenden linearen Zielfunktion, einem System von Gleichungs- bzw. Ungleichungsnebenbedingungen und unteren bzw. oberen Schranken für die Werte der (kontinuierlichen) Variablen. Zur Speicherung solcher Optimierungsprobleme verwendet man das sogenannte *LP-Format*. Eine Datei in diesem Format ist wie folgt aufgebaut (Leerzeilen werden ignoriert):

- *Problemname (optional)* Die erste Zeile kann den Namen des Problems enthalten. Sie beginnt dann mit "`\Problem name:`" gefolgt vom Namen des Problems.

  ```
  \Problem name: afiro.lp
  ```

- *Zielfunktion* Zunächst gibt es eine Zeile mit dem Eintrag "`Maximize`", "`Minimize`", "`Max`" oder "`Min`" (bzw. auch "`maximize`", etc.), die angibt, ob maximiert oder minimiert werden soll. Ab der nächsten Zeile folgt die Zielfunktion als Folge von Variablen mit Koeffizienten. Die Funktion beginnt mit einem Namen, der der Zielfunktion vorangestellt und mit einem Doppelpunkt abgeschlossen wird. Fehlt ein Koeffizient vor einer Variablen, so wird er als `1.0` angenommen. Es sind alle Variablennamen erlaubt (auch Zahlen!).

  ```
  Minimize
   COST: - 0.4 X02 - X14 - 0.6 Y123 - 0.48 X36 + 10 X39
  ```

- *Nebenbedingungen* Die Spezifikation der Nebenbedingungen beginnt mit einer Zeile, die "`Subject To`" oder "`s.t.`" enthält. Dann folgen die Nebenbedingungen. Die Nebenbedingungen sind analog zur Zielfunktion aufgebaut. Sie enthalten zusätzlich eine Angabe über die Art der Nebenbedingung ("`=`" für "=", "`<=`" für "≤" und "`>=`" für "≥") und die rechte Seite der Ungleichung als numerische Konstante. Die linke Seite der Ungleichung (bzw. Gleichung) kann auch leer sein.

  ```
  Subject To
   R10: - 1.06 X01 + X04  = 0
   X05: X01 <= 80
   X21: - X02 + 1.4 X14 >= 0
   10002A:  = 0
  ```

- *Schranken (optional)* Dieser Abschnitt beginnt mit einer Zeile "`Bounds`" oder "`bounds`". Schranken für eine Variable können auf die folgenden drei Arten angegeben werden:

  ```
  Bounds
  DMBOSONT >= -7
  DMBOSONT <= 0
  -3 <= DMBURYVR <= 0
  ```

  Variablen stehen also entweder auf der linken Seite oder in der Mitte. Variablen können auch auf einen festen Wert gesetzt werden:

  ```
   ULWP08 = 0
  ```

  Ferner kann auch der Fall auftreten, dass eine Variable unbeschränkt ist. Dies wird durch das Schlüsselwort `Free` gekennzeichnet:

  ```
   RGAP1 Free
  ```

  Ist für eine Variable keine unter Schranke angegeben, so wird diese als 0 angenommen. Ist keine obere Schranke angegeben, so wird diese als Unendlich angenommen.

- *Dateiende (optional)* Eine Zeile "`End`" oder "`end`" zeigt das Dateiende an.

Beschreiben Sie die Syntax einer Datei im LP-Format mit einer Grammatik in EBNF.

Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

2. Übungsblatt, Sommersemester 2015
Abgabetermin: 05.05.2015

Aufgabe 4

Überführen Sie die folgende Grammatik G in eine äquivalente Grammatik G' in Chomsky-Normalform.

$$S \to (\texttt{S})\texttt{S} \mid \varepsilon$$

Wenden Sie den Algorithmus für das Wortproblem aus der Vorlesung auf G' an und demonstrieren Sie dessen Ablauf mit den Eingabewörtern „`(())()`“ und „`((()(()`“.

Aufgabe 5

Entfernen Sie aus folgender Grammatik mit Startsymbol A alle direkten und indirekten Linksrekursionen:

$$A \to BC \mid \texttt{a}$$
$$B \to CD \mid D\texttt{b}$$
$$C \to \texttt{c}$$
$$D \to A\texttt{a}$$

Aufgabe 6

In der Grammatik für die Programmiersprache C wird der Aufzählungstyp mit den folgenden Regeln spezifiziert:

enum-specifier:
- `enum` *identifier*$_{opt}$ **{** *enumerator-list* **}**
- `enum` *identifier*

enumerator-list:
- *enumerator*
- *enumerator-list* **,** *enumerator*

enumerator:
- *identifier*
- *identifier* = *constant-expression*

Die Schreibweise ist folgendermaßen zu verstehen: Die *kursiv* geschriebenen Worte stellen Nichtterminale dar. Die in `Schreibmaschinenschrift` geschriebenen Worte sind Terminalsymbole, insbesondere „**{** “, „**}**“, „**,**“und „=“. Weiterhin stellt jeder übergeordnete Begriff gefolgt von einem Doppelpunkt die linke Seite einer Produktion und die untergeordneten Zeilen Alternativen der rechten Seite dar. Ein optionales Symbol ist mit dem Index “opt” markiert.

Schreiben Sie diese Regeln in der erweiterten Backus-Naur-Form (EBNF) auf und zeichnen Sie das dazugehörige Syntaxdiagramm.

– Bitte wenden –

Aufgabe 7

Geben Sie zu jeder der folgenden regulären Sprachen über dem Alphabet $\{\mathbf{0}, \mathbf{1}\}$ einen deterministischen endlichen Automaten an.

a) $\{w \mid w$ beginnt mit $\mathbf{0}$ und hat eine ungerade Länge oder beginnt mit $\mathbf{1}$ und hat eine gerade Länge$\}$,

b) $\{w \mid w$ enthält den Teilstring $\mathbf{0101}\}$.

Prof. Dr. Gerhard Reinelt
Dipl.-Math. Stefan Wiesberg
Institut für Informatik
Universität Heidelberg

Compilerbau

1. Übungsblatt, Sommersemester 2015
Abgabetermin: 28.04.2015

Aufgabe 1

Betrachten Sie die kontextfreie Grammatik

$$S \to SS+ \mid SS* \mid a$$

a) Wie kann das Wort $aa + a*$ erzeugt werden?

b) Konstruieren Sie den Ableitungsbaum für dieses Wort.

c) Welche Sprache wird von dieser Grammatik erzeugt? Begründen Sie Ihre Antwort.

Aufgabe 2

Welche Sprache wird jeweils von den folgenden Grammatiken erzeugt? Begründen Sie Ihre Antworten.

a) $S \to 0S1 \mid 01$

b) $S \to +SS \mid -SS \mid a$

c) $S \to S(S)S \mid \varepsilon$

d) $S \to aSbS \mid bSaS \mid \varepsilon$

e) $S \to a \mid S + S \mid SS \mid S * S \mid (S)$

Welche dieser Grammatiken sind mehrdeutig?

Aufgabe 3

a) Zeigen Sie, dass alle Binärzahlen, die von der folgenden Grammatik erzeugt werden, durch 3 teilbar sind. Tipp: Induktion über die Anzahl der Knoten im Ableitungsbaum.

$$S \to 11 \mid 1001 \mid S0 \mid SS$$

b) Erzeugt diese Grammatik *alle* positiven durch 3 teilbaren Binärzahlen?