FCW 1x Übung zu Formale Sprachen, Compiler- und Werkzeugbau 1 — WS 2016/17, Übung 2

Abgabetermin: in der KW 45

☐ **Gr. 1,** Dr. H. Dobler
☐ **Gr. 2,** Dr. G. Kronberger

Name _____ **Aufwand in h** _____

Punkte _____ **Übungsleiter** _____

1. Kanonische Ableitung und Reduktion (2 + 3 Punkte)

Gegeben sei folgende Grammatik für einfache arithmetische Ausdrücke:

```
E → T | + T | - T | E + T | E - T
T → F |             T * F | T / F
F → v | ( E )
```

a) Leiten Sie aus dem Satzsymbol *E* einmal links- und einmal rechtskanonisch folgenden Satz ab:

```
- v * ( v + v / v )
```

b) Reduzieren Sie folgenden Satz

```
( ( v + v ) * v / v ) – ( v / v )
```

einmal links- und einmal rechtskanonisch bis zum Satzsymbol und kennzeichnen Sie in jeder Satzform den Ansatz durch Unterstreichen. Zeichnen Sie dann den Syntaxbaum für diesen Satz und vergleichen Sie ihn mit den beiden Ableitungsfolgen. Was stellen Sie dabei fest?

2. Mehrdeutigkeit, Beschreibung und Schreibweisen (1 + 2 Punkte)

Eine Programmiersprache erlaubt Fließkomma-Literale, deren Syntax durch folgende Grammatik *G*(*real*) beschrieben werden kann:

```
real →  mant | mant exp
mant →  sign int | sign int . frac
int  →  n | int n
frac →  n | frac n | ε
sign →  + | - | ε
exp  →  E sign int
n    →  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

a) Ist diese Grammatik mehrdeutig? Begründen Sie Ihre Antwort und transformieren Sie diese Grammatik – wenn nötig – in eine äquivalente eindeutige Grammatik.

b) Geben Sie eine möglichst kurze Grammatik für *L*(*G*(*real*)) in Wirth'scher EBNF an.

3. Reguläre Grammatiken (2 + 2 Punkte)

Gegeben sei folgende reguläre Grammatik $G(S)$:

$S \rightarrow bA \mid aB \mid \varepsilon$
$A \rightarrow bA \mid b$
$B \rightarrow bC \mid b$
$C \rightarrow aB$

a) Geben Sie eine äquivalente *umgekehrt reguläre Grammatik* an.

b) Geben Sie einen *regulären Ausdruck* an, der die Sprache dieser Grammatik beschreibt.

4. Bezeichner in der Programmiersprache Ada (2 + 2 + 2 Punkte)

Bezeichner der Programmiersprache Ada dürfen aus Buchstaben (Terminalklasse *l*), Ziffern (Terminalklasse *d*) und dem Unterstreichungszeichen ('_') bestehen. Ada-Bezeichner müssen mit einem Buchstaben beginnen und dürfen nicht mit einem Unterstreichungszeichen enden. Dazwischen dürfen keine zwei Unterstreichungszeichen unmittelbar hintereinander vorkommen.

a) Geben Sie eine *reguläre Grammatik* an, die diese Bezeichner beschreibt.

b) Geben Sie eine *umgekehrt reguläre Grammatik* an, die diese Bezeichner beschreibt.

c) Leiten Sie aus der regulären Grammatik einen *regulären Ausdruck* ab, der die Menge aller Bezeichner beschreibt. Kann man den so hergeleiteten Ausdruck noch verkürzen?

5. Transformation der Darstellungsformen regulärer Sprachen (3 + 3 Punkte)

a) Eine reguläre Sprache sei durch folgenden *regulären Ausdruck* definiert:

$(ab)^*(ba)^* + aa^*$

Konstruieren Sie für diese Sprache einen *deterministischen endlichen Automaten*.

Hinweis: Gehen Sie dabei so vor, dass Sie aus dem regulären Ausdruck zuerst einen *nichtdeterministischen* Automaten "ablesen" und diesen dann in einen *deterministischen* transformieren.

b) Die Menge aller Ketten aus 0 und 1, die als Dual- oder Binärzahl interpretiert, ohne Rest durch 3 teilbar sind, bildet eine *reguläre Sprache*. Geben Sie für diese Sprache einen *deterministischen endlichen Automaten* (mit Erläuterungen!) an.

Hinweis: Überlegen Sie für eine beliebige Dualzahl x, welche Auswirkungen auf den Divisionsrest (x *mod* 3) das Anhängen von 0 (neue Zahl $x0$) bzw. von 1 (neue Zahl $x1$) hat.

1 Kanonische Ableitung und Reduktion

Dieser Abschnitt behandelt die Aufgabe 1 der zweiten Übung.

1.1 Rechtskanonische Ableitung

Folgende Ableitung beweist, dass der Satz $-v * (v + v/v)$ ein Satz der Sprache ist, die durch die gegebene Grammatik definiert ist.

$$
\begin{aligned}
E &\overset{L}{\Rightarrow} \underline{-T} \\
&\overset{L}{\Rightarrow} \underline{-T} * F \\
&\overset{L}{\Rightarrow} \underline{-F} * F \\
&\overset{L}{\Rightarrow} -v * \underline{F} \\
&\overset{L}{\Rightarrow} -v * (\underline{E}) \\
&\overset{L}{\Rightarrow} -v * (\underline{E} + T) \\
&\overset{L}{\Rightarrow} -v * (\underline{T} + T) \\
&\overset{L}{\Rightarrow} -v * (\underline{F} + T) \\
&\overset{L}{\Rightarrow} -v * (v + \underline{T}) \\
&\overset{L}{\Rightarrow} -v * (v + \underline{T}/F) \\
&\overset{L}{\Rightarrow} -v * (v + \underline{F}/F) \\
&\overset{L}{\Rightarrow} -v * (v + v/\underline{F}) \\
&\overset{L}{\Rightarrow} -v * (v + v/v)
\end{aligned}
$$

1.2 Kanonische Reduktion

Folgende Reduktion beweist das der Satz $((v + v) * v/v) - (v/v)$ ein Satz der Sprache ist, die durch die vorgegebene Grammatik definiert ist.

1.2.1 Linkskanonische Reduktion

$$
\begin{aligned}
\vdash^{L} \quad & ((\underline{v} + v) * v/v) - (v/v) \\
\vdash^{L} \quad & ((\underline{F} + v) * v/v) - (v/v) \\
\vdash^{L} \quad & ((\underline{T} + v) * v/v) - (v/v) \\
\vdash^{L} \quad & ((E + \underline{v}) * v/v) - (v/v) \\
\vdash^{L} \quad & ((E + \underline{F}) * v/v) - (v/v) \\
\vdash^{L} \quad & (\underline{(E + T)} * v/v) - (v/v) \\
\vdash^{L} \quad & (\underline{(E)} * v/v) - (v/v) \\
\vdash^{L} \quad & (\underline{F} * v/v) - (v/v) \\
\vdash^{L} \quad & (T * \underline{v}/v) - (v/v) \\
\vdash^{L} \quad & (\underline{T * F}/v) - (v/v) \\
\vdash^{L} \quad & (T/\underline{v}) - (v/v) \\
\vdash^{L} \quad & \overline{(T/F)} - (v/v) \\
\vdash^{L} \quad & (\underline{T}) - (v/v) \\
\vdash^{L} \quad & (\underline{E}) - (v/v) \\
\vdash^{L} \quad & \underline{F} - (v/v) \\
\vdash^{L} \quad & \underline{T} - (v/v) \\
\vdash^{L} \quad & E - (\underline{v}/v) \\
\vdash^{L} \quad & E - (\underline{F}/v) \\
\vdash^{L} \quad & E - (\underline{T}/v) \\
\vdash^{L} \quad & E - \overline{(T/F)} \\
\vdash^{L} \quad & E - (\underline{T}) \\
\vdash^{L} \quad & E - (\underline{E}) \\
\vdash^{L} \quad & E - \underline{F} \\
\vdash^{L} \quad & \underline{E - T} \\
\vdash^{L} \quad & E
\end{aligned}
$$

1.2.2 Rechtskanonische Reduktion

$$
\begin{aligned}
&\vdash^R \quad ((v+v)*v/v)-(v/\underline{v}) \\
&\vdash^R \quad ((v+v)*v/v)-(\underline{v}/F) \\
&\vdash^R \quad ((v+v)*v/v)-(\underline{F}/F) \\
&\vdash^R \quad ((v+v)*v/v)-\underline{(T/F)} \\
&\vdash^R \quad ((v+v)*v/v)-(\underline{T}) \\
&\vdash^R \quad ((v+v)*v/v)-\underline{(E)} \\
&\vdash^R \quad ((v+v)*v/v)-\underline{F} \\
&\vdash^R \quad ((v+v)*v/\underline{v})-T \\
&\vdash^R \quad ((v+v)*\underline{v}/F)-T \\
&\vdash^R \quad ((v+\underline{v})*F/F)-T \\
&\vdash^R \quad ((v+\underline{F})*F/F)-T \\
&\vdash^R \quad ((\underline{v}+T)*F/F)-T \\
&\vdash^R \quad ((\underline{F}+T)*F/F)-T \\
&\vdash^R \quad ((\underline{T}+T)*F/F)-T \\
&\vdash^R \quad ((\underline{E+T})*F/F)-T \\
&\vdash^R \quad ((\underline{E})*F/F)-T \\
&\vdash^R \quad (\underline{F}*F/F)-T \\
&\vdash^R \quad (\underline{T*F}/F)-T \\
&\vdash^R \quad \underline{(T/F)}-T \\
&\vdash^R \quad (\underline{T})-T \\
&\vdash^R \quad (\underline{E})-T \\
&\vdash^R \quad \underline{F}-T \\
&\vdash^R \quad \underline{T}-T \\
&\vdash^R \quad \underline{E-T} \\
&\vdash^R \quad E
\end{aligned}
$$

2 Mehrdeutigkeit, Beschreibung und Schreibweise

2.1 Mehrdeutigkeit der Grammatik

Folgendes Beispiel zeigt das die vorgegebene Grammatik für den Satz +1.5 mehrdeutig ist, was durch die Regel $frac \rightarrow \epsilon | fractn$ verursacht wird.

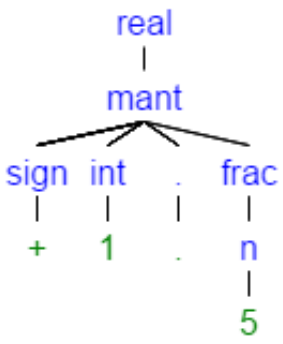

Abbildung 1: Syntaxbaum 1

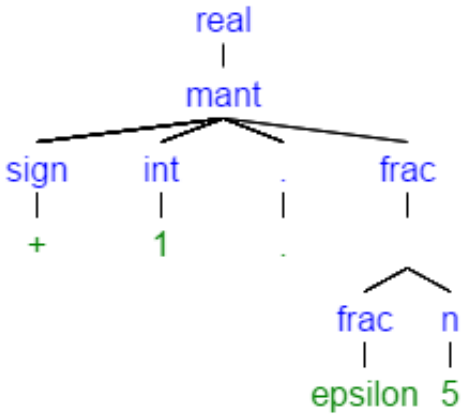

Abbildung 2: Syntaxbaum 2

real	⇒	mant | mant exp
mant	⇒	sign int | sign int . frac
int	⇒	n | int n
frac	⇒	n | frac n
sign	⇒	+ | -
exp	⇒	E sign int
n	⇒	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ϵ

2.2 Grammatik mit Wirth'scher Schreibweise

Folgende Grammatik ist die vorgegebene Grammatik in der Wirth'schen Schreibweise.

G = [“+” | “+”] n {n} [“.” {n}] [“E” [“+” | “+”] n {n}].

3 Reguläre Sprachen

3.1 Reguläre Grammatik

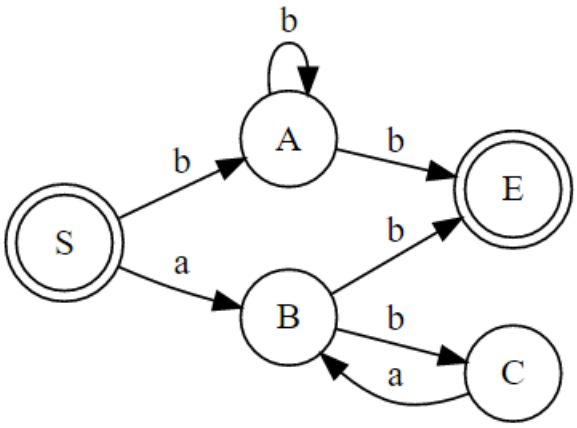

Abbildung 3: Automat

Folgende Grammatik wurde aus dem Automaten abgeleitet. Diese Grammatik muss aber noch angepasst werden, denn die Regel S $\rightarrow \epsilon$ ist in einer regulären Grammatik nicht gültig.

Die folgende Grammatik

E	$\rightarrow$	Ab | Bb
A	$\rightarrow$	Ab | ~~S~~b
B	$\rightarrow$	~~S~~a | Ca
C	$\rightarrow$	Bb
~~S~~	$\rightarrow$	~~ϵ~~

wird umgeformt zu

E	$\rightarrow$	Ab | Bb
A	$\rightarrow$	Ab | b
B	$\rightarrow$	a | Ca
C	$\rightarrow$	Bb

3.2 Umgekehrte reguläre Grammatik

Die folgende Auflistung zeigt die drei regulären Ausdrücke, welche die Grammatik aus der Aufgabenstellung 3 beschreiben.

1. $\epsilon+$
2. $bb*b+$
3. $a(ba)*b$

3.3 Regulärer Ausdruck der Grammatik

Folgender regulärer Ausdruck beschreibt die Grammatik aus der Aufgabenstellung 3.

$L = \epsilon + bb*b + a(ba)*b$

4 Bezeichner in der Programmiersprache Ada

Dieser Abschnitt beschäftigt sich mit der Aufgabenstellung 4 der dritten Übung.

4.1 Reguälre Grammatik

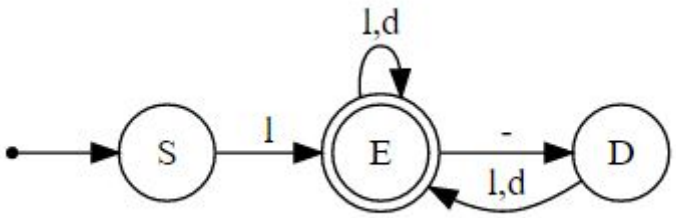

Abbildung 4: Ada reguläre Grammatik

S → lE | l
E → lE | dE | "_"U | l | d
U → lE | dE | l | d

4.2 Umgekehrte reguläre Grammatik

E → El | Ed | Ul | Ud | l
U → E"_"

4.3 Regulärer Ausdruck

$l((l+d)+(\_(l+d))*$

5 Transformation der Darstellungsformen der regulären Sprachen

5.1 Reguälre Grammatik

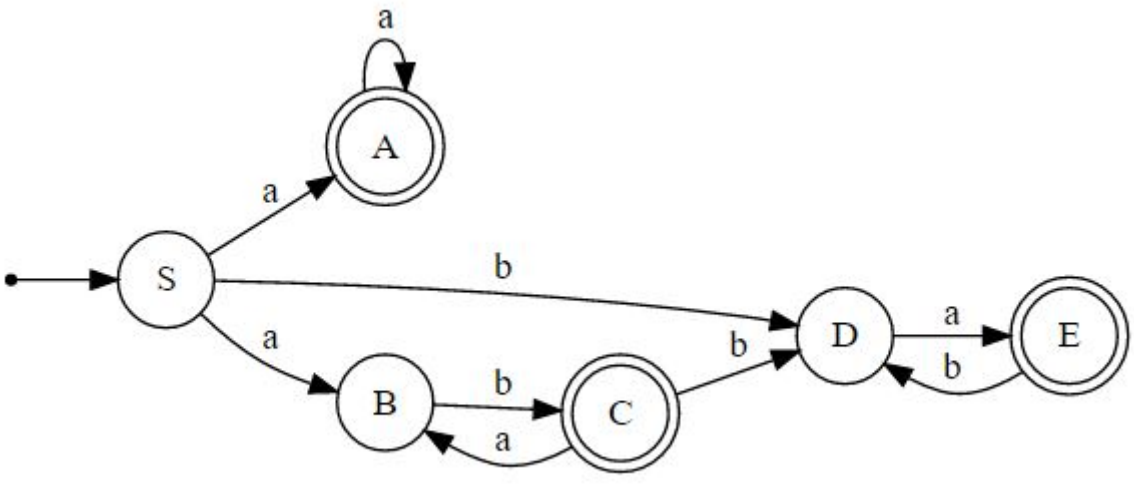

Abbildung 5: Nicht deterministischer Zustandsautomat des transformierten Ausdrucks

Zustandsüberführungsfunktion

	a	b
S	{A,B}	{D}
A	{A}	{}
B	{}	{C}
C	{B}	{D}
D	{E}	{}
E	{}	{D}

transformierte Zustandsüberführungsfunktion

	a	b
S	{A,B}	{D}
{A,B}	{A}	{C}
A	{A}	{}
B	{}	{C}
C	{B}	{D}
D	{E}	{}
E	{}	{D}

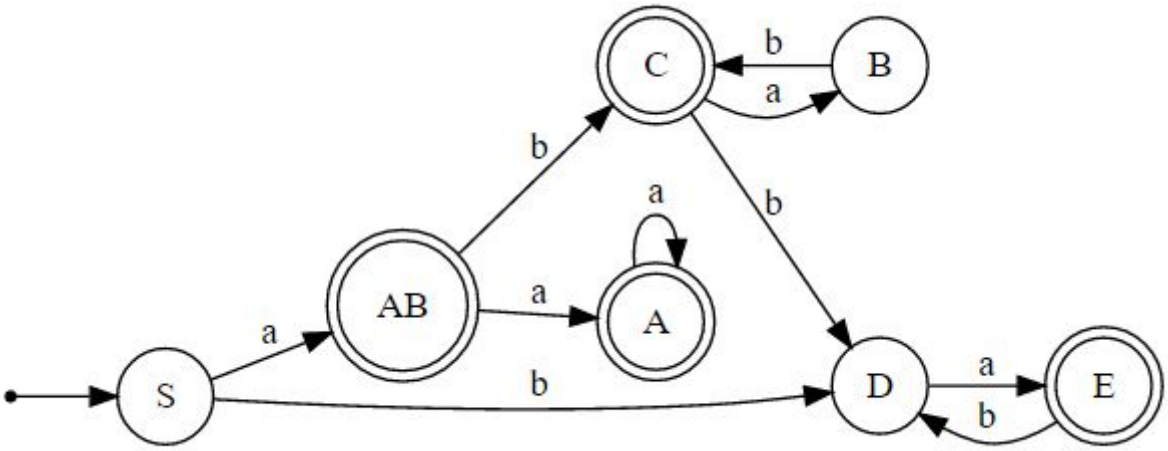

Abbildung 6: Deterministischer Zustandsautomat des transformierten Ausdrucks