Praktischer Zettel 1 Aufgabe 1

Aufgabe a)

Ausgangsgrammatik

Für die Grammatik bin ich von dem Schema in den Folien ausgegangen und habe zunächst die fehlenden Operatoren ergänzt, was folgende Grammatik ergibt:

$$
\begin{aligned}
&S \rightarrow E\# \\
&E \rightarrow E + P \quad | \quad E - P \quad | \quad P \\
&P \rightarrow P * T \quad | \quad P/T \quad | \quad T \\
&T \rightarrow T \,\hat{}\, F \quad | \quad F \\
&F \rightarrow (E) \quad | \quad Z
\end{aligned}
$$

Die Reihenfolge der Nichtterminale ist wichtig, da dies die Präzendenzregeln zur Folge hat. Z.B. muss bei einem + erst das P ausgewertet werden, d.h. der Produktterm hat höhere Priorität. Analog mit dem Potenzterm.

Dies Grammatik hat jetzt noch diverse Probleme, die wir nachfolgend beheben werden. Das Ziel ist dabei, eine $LL(1)$-Grammatik zu erhalten, da eine Parser mit rekursivem Abstieg für diese garantiert terminiert und lineare Laufzeit hat.

Das Nichtterminal Z steht für eine positive Zahl. Eine Grammatik dafür sieht wie folgt aus:

$$
\begin{aligned}
&Z \rightarrow 0 \quad | \quad 1A \quad | \quad ... \quad | \quad 9A \quad | \quad .B \\
&A \rightarrow \varepsilon \quad | \quad 1A \quad | \quad ... \quad | \quad 9A \quad | \quad .B \\
&B \rightarrow 0C \quad | \quad ... \quad | \quad 9C \\
&C \rightarrow 0C \quad | \quad ... \quad | \quad 9C \quad | \quad \varepsilon
\end{aligned}
$$

Linksrekursion entfernen

Der erste Schritt ist die Linksrekursion zu entfernen, analog zu den Folien, somit erhält man:

$$\begin{aligned}
S &\to E\# \\
E &\to PE' \\
E' &\to +PE' \quad | \quad -PE' \quad | \quad \varepsilon \\
P &\to TP' \\
P' &\to *TP' \quad | \quad /TP' \quad | \quad \varepsilon \\
T &\to FT' \\
T' &\to \char`^FT' \quad | \quad \varepsilon \\
F &\to (E) \quad | \quad Z
\end{aligned}$$

Rechtsassoziativität vom Potenz-Operator ermöglichen

Als nächstes wollen wir den Potenzoperator rechtsassoziativ machen, d.h. dass $a \char`^ b \char`^ c = a \char`^ (b \char`^ c)$ und nicht $a \char`^ b \char`^ c = (a \char`^ b) \char`^ c$. Man könnte Terme mit mehreren Potenzoperatoren (ohne Klammern auch verbieten), indem man die Regel T' zu $T' \to \char`^F$ anpasst.

Der Grund, warum der Operator aktuell linksassoziativ ist, ist dass das F in der Regel $T' \to \char`^FT'$ zuerst ausgewertet wird (leftmost evaluation) und dann direkt mit dem Operator verknüpft wird. Wir müssten erreichen, dass ganz $FT'$ ausgewertet wird. Das geht also durch Anpassen der Regel zu $T' \to \char`^T'$.

Ingesamt ergibt sich so dann:

$$\begin{aligned}
S &\to E\# \\
E &\to PE' \\
E' &\to +PE' \quad | \quad -PE' \quad | \quad \varepsilon \\
P &\to TP' \\
P' &\to *TP' \quad | \quad /TP' \quad | \quad \varepsilon \\
T &\to FT' \\
T' &\to \char`^T \quad | \quad \varepsilon \\
F &\to (E) \quad | \quad Z
\end{aligned}$$

Unäres Minus ermöglichen

Als letztes möchten wir das unäre Minus ermöglichen. Das ist auch nötig, damit negative Zahlen möglich sind. Es gibt die Möglichkeit, die Regeln E, P, T und F anzupassen:

- Das Minus bei der Regel F (z.B durch Hinzufügen von $F \rightarrow -Z$) würde ein Problem bei Termen der Form $-a \,\hat{}\, b$ verursachen. Das Minus hätte jetzt eine höhere Priorität als die Potenz, was zum Effekt hat, dass der Term äquivalent zu $(-a) \,\hat{}\, b$ ist, dies ist aber inkorrekt.
- Das Minus darf ebenfalls nicht in der E-Regel vorkommen (z.B. via $E \rightarrow -E$), denn das hätte zum Effekt, dass es eine höhere Priorität als Plus und Minus hat, z.B. wäre $-a + b$ dann äquivalent zu $-(a + b)$, was falsch ist.
- Das Minus könnte prinzipiell in der P-Regel vorkommen, das Problem ist aber, dass wir von P' nicht mehr zu P "zurück" kommen und daher wäre ein Minus beim rechten Operand nicht möglich, z.B. im Falle von $4 * -3$.

Für die Regel T würde vor der Anpassung der Rechtsassoziativität dasselbe Problem gelten wie bei P. Man könnte z.B. zwei F-Regeln machen, eine die negative Zahlen erlaubt und eine die es nicht tut und dann die T-Regel anpassen (z.B. zu $T \rightarrow FT'|F'$ wobei F nur positive und F' auch negative Zahlen erlaubt).

Da wir aber die Rechtsassoziativität angepasst haben, kommt man von T' zu T zurück. D.h. wir können die Regel T einfach wie folgt ergänzen: $T \rightarrow -T$. Hier kann also auch korrekterweise den rechten Operand negativ haben.

Somit ergibt sich:

$$
\begin{aligned}
S &\rightarrow E\# \\
E &\rightarrow PE' \\
E' &\rightarrow +PE' \quad | \quad -PE' \quad | \quad \varepsilon \\
P &\rightarrow TP' \\
P' &\rightarrow *TP' \quad | \quad /TP' \quad | \quad \varepsilon \\
T &\rightarrow -T \quad | \quad FT' \\
T' &\rightarrow \hat{}\,T \quad | \quad \varepsilon \\
F &\rightarrow (E) \quad | \quad Z
\end{aligned}
$$

Gesamtergebnis

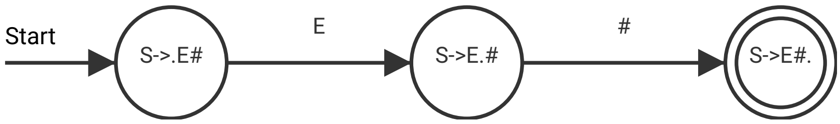
Die Grammatik ist nun fertig, prüft man die FIRST- und FOLLOW-Mengen, so wird man feststellen, dass sie in der Tat das $LL(1)$-Kriterium erfüllt und daher perfekt für einen Parser mit rekursivem Abstieg geeignet ist.

Hier nochmal die Grammatik:

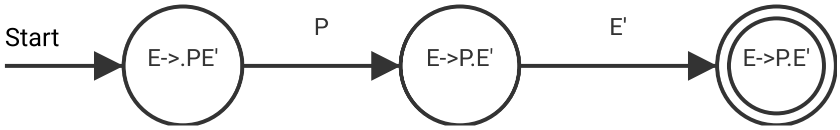
$$
\begin{aligned}
S &\to E\# \\
E &\to PE' \\
E' &\to +PE' \quad | \quad -PE' \quad | \quad \varepsilon \\
P &\to TP' \\
P' &\to *TP' \quad | \quad /TP' \quad | \quad \varepsilon \\
T &\to -T \quad | \quad FT' \\
T' &\to \hat{\ }T \quad | \quad \varepsilon \\
F &\to (E) \quad | \quad Z
\end{aligned}
$$

Aufgabe b)

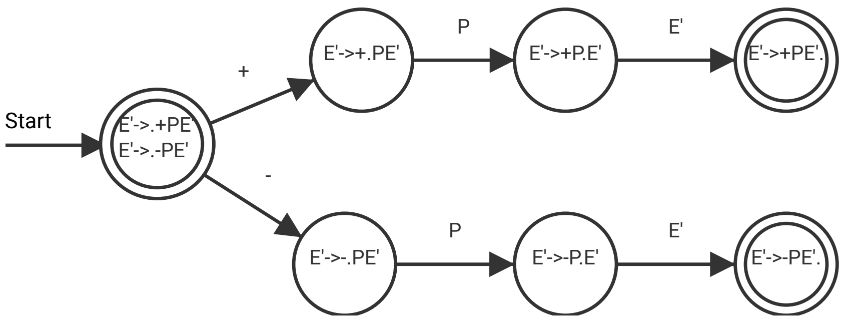
Automat für S:



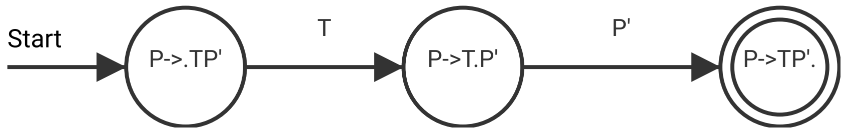
Automat für E:



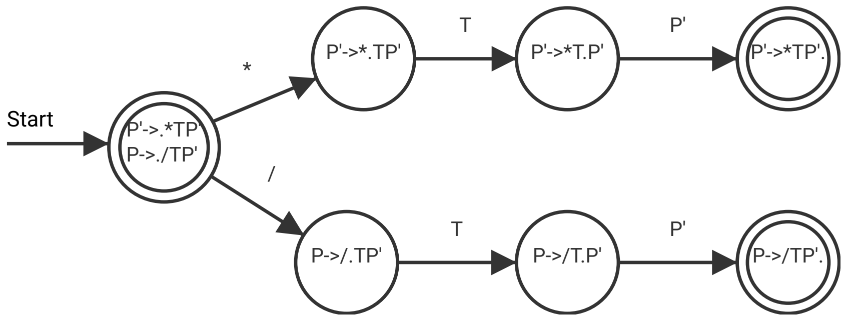
Automat für E':



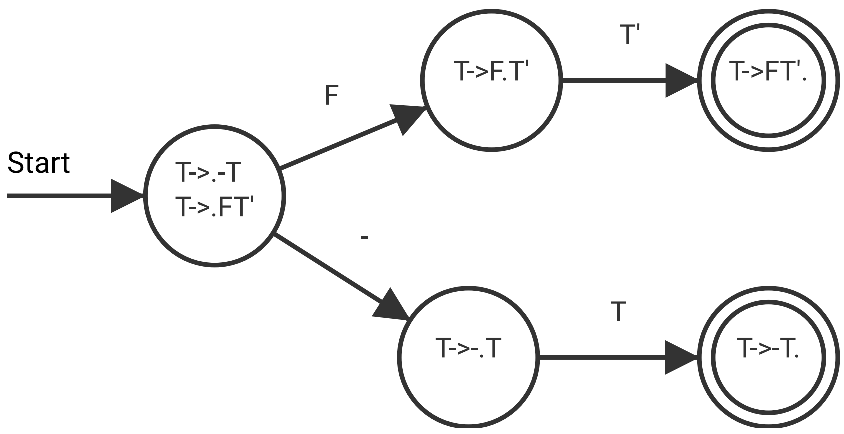
Automat für P:



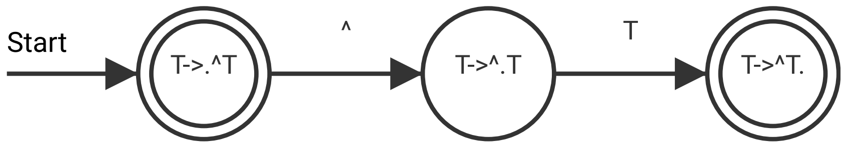
Automat für P':



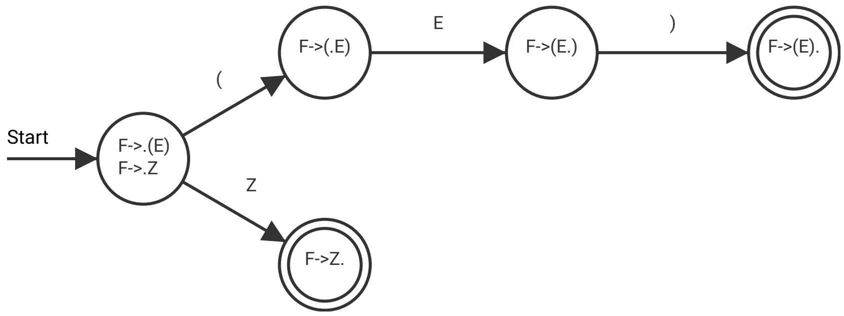
Automat für T:



Automat für T':



Automat für F:



Der Automat für Z wurde ausgelassen, da die Handhabung aller 10 Fälle für die unterschiedlichen Ziffern die Automaten sehr unübersichtlich aussehen ließe und die Automaten sowieso trivial sind.

Aufgabe b)

Zunächst die Konstruktion wie im Skript mit ε-Übergängen

Entfernung der ε-Übergänge liefert den DEA ohne Fehlerzustand (zur Übersichtlichkeit):

Aufgabe c)

Mögliche Konfikte sind:

- *reduce/reduce*-Konflikte: Ein Endzustand hat mehr als eine komplett gelesene Produktion. Solche Endzustände gibt es nicht
- *shifte/reduce*-Konflikte: Ein Endzustand hat eine komplett gelesene Produktion und eine nicht komplett gelesene Produktion mit einem Terminalsymbol nach dem Punkt. Es gibt einen solchen Endzustand: Den Zustand mit $B \to b.$, $B \to .cA$.

Aufgabe d)

$SLR(1)$-Konflikte, liegen vor falls:

- Es gibt einen Zustand zwei unterschiedlichen reduce-Items, die $LA([A \to \alpha.]) \cap LA([B \to \beta.]) \neq \emptyset$ erfüllen. Da wir keine *reduce/reduce*-Konflikte haben ist dies nicht der Fall.
- Für zwei Items $[A \to \alpha.]$ und $[B \to \alpha.a\beta]$ ist $a \in LA([A \to \alpha])$. Der obige Zustand mit dem *shift/reduce*-Konflikt erfüllt dies nicht, denn: $b \notin FOLLOW_1(B) = \{a\}$

Somit ist dies auf jeden Fall eine $SLR(1)$-Grammatik. Der DEA sieht wie folgt aus:

Kritisch war ja nur der Zustand $q6$. Mit einem SLR-Parser ist dies nun wie folgt zu verstehen: Wenn das Lookahead-Zeichen im Follow-Set von B liegt, d.h. das Zeichen ist ein a, dann wende die Reduktion an, von dieser gibt es ja nur eine ($B \to b$).
In allen anderen Fällen die entsprechende Shift-Operation. Reduktionen sind rot markiert, Shifts grün.

Aufgabe e)

Vorgehen lt. Folie: alle Zustände im Automaten zu Endzuständen machen. Die akzeptierten Wörter sind genau die zuverlässigen Präfixe.

- ab: Ja, Items: $S \to .aBa$, $B \to .b$, $B \to b.$
- ba: Nein, jeder Satz muss mit a beginnen ($FIRST_1(S) = \{a\}$)
- acb: Nein, nach c muss a folgen ($FOLLOW_1(c) = \{a\}$)
- abc: Ja, Items: $S \to .aBa$, $B \to .bB$, $B \to .cA$, $B \to c.A$
- aba: Nein, auf b folgt niemals a, nur b oder c ($FOLLOW_1(b) = \{b, c\}$)

Aufgabe f)

q_i	Aktion (l steht für lookahead)
q_0	*shift*
q_1	*accept*
q_2	*shift*
q_3	*shift*
q_4	*reduce($A \to aBa$)*
q_5	*reduce($B \to b$)* falls $l = a$, sonst *shift*
q_6	*reduce($B \to bB$)*
q_7	*shift*
q_8	*reduce($B \to cA$)*
q_9	*error*

	a	b	c	A	B
q_0	q_2			q_1	
q_1					
q_2		q_5	q_7		q_3
q_3	q_4				
q_4					
q_5		q_5	q_7		q_6
q_6					
q_7	q_2			q_8	
q_8					
q_9					

Hierbei steht q_9 für den im Automaten aus Gründen der Übersichtlichkeit nicht vorhandenen Fehlerzustand.

Dies lässt sich auch mit nur einer Tabelle darstellen:

- Eine zusätzliche Spalte *eof* für das Eingabe-Ende. Die Follow-Mengen der Nichtterminale müssen dies berücksichtigen (z.B. gilt dann $FOLLOW_1(S) = \{eof\}$ u. $FOLLOW_1(A) = \{eof, a\}$)
- Die Spalten (außer der ersten natürlich) stehen jeweils fürdas Lookahead-Zeichen
- Für Shift-Aktionen enthält die Zelle den Nachfolgezustand
- Im Falle einer Reduktion kommt die Reduktions-Aktion nur in die Spalten, deren Lookahead-Zeichen in der Follow-Menge liegt.
- Die accept-Aktion kommt in die *eof*-Spalte für die Zeile des Endzustands des Item-Automaten des Startsymbols
- Der Rest wird mit der Fehleraktion gefüllt

So ergibt sich die folgende Tabelle:

q_i	*eof*	a	b	c	A	B
q_0	err	q_2	err	err	q_1	err
q_1	**acc**	err	err	err	err	err
q_2	err	err	q_5	q_7	err	q_3
q_3	err	q_4	err	err	err	err
q_4	$r(A \to aBa)$	$r(A \to aBa)$	err	err	err	err
q_5	err	$r(B \to b)$	q_5	q_7	err	q_6
q_6	err	$r(B \to bB)$	err	err	err	err
q_7	err	q_2	err	err	q_8	err
q_8	err	$r(B \to cA)$	err	err	err	err

Aufgabe g)

$$S \Longrightarrow aBa \Longrightarrow abBa \Longrightarrow abbBa \Longrightarrow abbcAa \Longrightarrow abbcaBaa \Longrightarrow abbcabaa$$

Stack	Input Rest	Aktion
q_0	$abbcabaa$	*shift*: q_2
$q_0\ q_2$	$bbcabaa$	*shift*: q_5
$q_0\ q_2\ q_5$	$bcabaa$	*shift*: q_5
$q_0\ q_2\ q_5\ q_5$	$cabaa$	*shift*: q_7
$q_0\ q_2\ q_5\ q_5\ q_7$	$abaa$	*shift*: q_2
$q_0\ q_2\ q_5\ q_5\ q_7\ q_2$	baa	*shift*: q_5
$q_0\ q_2\ q_5\ q_5\ q_7\ q_2\ q_5$	aa	$r(B \to b)$
$q_0\ q_2\ q_5\ q_5\ q_7\ q_2\ q_3$	aa	*shift*: q_4
$q_0\ q_2\ q_5\ q_5\ q_7\ q_2\ q_3\ q_4$	a	$r(A \to aBa)$
$q_0\ q_2\ q_5\ q_5\ q_7\ q_8$	a	$r(B \to cA)$
$q_0\ q_2\ q_5\ q_5\ q_6$	a	$r(B \to bB)$
$q_0\ q_2\ q_5\ q_6$	a	$r(B \to bB)$
$q_0\ q_2\ q_3$	a	*shift*: q_4
$q_0\ q_2\ q_3\ q_4$	*eof*	$r(A \to aBa)$
$q_0\ q_1$	*eof*	**acc**