Prof. Dr. M. Müller C. Aschermann, J. Hensel, J. Protze, P. Reble

Tutoraufgabe 1 (Syntax und Semantik):

a) Die Menge der syntaktisch korrekten einfachen arithmetischen Ausdrücke (**EAA**) wird durch die Grammatik $G_1 = (\{S_1\}, \{(,), ;, \mathsf{plus}, \mathsf{s}, \mathcal{O}\}, P_1, S_1)$ definiert, wobei P_1 genau die folgenden Produktionen enthält:

$$\begin{aligned} S_1 &\to \mathcal{O} \\ S_1 &\to \mathsf{s}(S_1) \\ S_1 &\to \mathsf{plus}(S_1; S_1) \end{aligned}$$

Die Semantik $\mathcal{W}(\mathcal{A})$ eines syntaktisch korrekten **EAA**s $\mathcal{A}$ ist wie folgt definiert, wobei x und y ebenfalls syntaktisch korrekte **EAA**s sind:

$$\begin{aligned} \mathcal{W}(\mathcal{O}) &= 0 \\ \mathcal{W}(\mathsf{s}(x)) &= \mathcal{W}(x) + 1 \\ \mathcal{W}(\mathsf{plus}(x; y)) &= \mathcal{W}(x) + \mathcal{W}(y) \end{aligned}$$

Für alle **EAA**s $\mathcal{A}$ gilt also $\mathcal{W}(\mathcal{A}) \in \mathbb{N}$.

Geben Sie für die folgenden drei Ausdrücke an, ob es sich um einen syntaktisch korrekten **EAA** handelt und welche Semantik er hat.

i) $\mathsf{plus}(\mathsf{s}(\mathcal{O}); \mathcal{O})$ ii) $\mathsf{plus}(\mathcal{O}; \mathsf{s}(\mathcal{O}); \mathcal{O})$ iii) $\mathsf{plus}(\mathsf{plus}(\mathcal{O}; \mathsf{s}(\mathcal{O})); \mathsf{s}(\mathsf{plus}(\mathcal{O}; \mathcal{O})))$

b) Beweisen oder widerlegen Sie, dass im Allgemeinen gilt: Zwei Ausdrücke mit gleicher Syntax haben auch die gleiche Semantik.

Lösung:

a) i) Der Ausdruck ist syntaktisch korrekt und seine Semantik ist 1.

ii) Der Ausdruck ist syntaktisch nicht korrekt und hat daher keine Semantik.

iii) Der Ausdruck ist syntaktisch korrekt und seine Semantik ist 2.

b) Die Aussage ist falsch. Betrachten wir den (nicht einfachen) arithmetischen Ausdruck `1 / 2`. In Java wird dieser Ausdruck zu 0 ausgewertet, da Java bei der Division zweier ganzer Zahlen die Ganzzahldivision ohne Rest verwendet. In Prolog wird dieser Ausdruck hingegen zu 0.5 ausgewertet, da Prolog für den Operator `/` grundsätzlich Fließkommadivision verwendet. Dieses Gegenbeispiel widerlegt die Aussage.

Aufgabe 2 (Syntax und Semantik): (4 + 1 + 1 = 6 Punkte)

a) Die Menge der syntaktisch korrekten **SASP** Programme wird durch die Grammatik $G_2 = (\{A, B, S_2\}, \{\texttt{.}, \texttt{:-}, \texttt{p}, \texttt{q}, \texttt{r}, \texttt{s}\}, P_2, S_2)$ definiert, wobei P_2 genau die folgenden Produktionen enthält:

$$\begin{aligned} S_2 &\to A\texttt{.} \\ S_2 &\to A\texttt{.}S_2 \\ A &\to B \\ A &\to B\texttt{:-}B \\ B &\to \texttt{p} \\ B &\to \texttt{q} \\ B &\to \texttt{r} \\ B &\to \texttt{s} \end{aligned}$$

Die Semantik $\mathcal{W}(\mathcal{P})$ eines syntaktisch korrekten **SASP** Programms $\mathcal{P}$ ist wie folgt definiert, wobei $\mathcal{P}'$ ebenfalls ein syntaktisch korrektes **SASP** Programm ist und $x, y \in \{\texttt{p}, \texttt{q}, \texttt{r}, \texttt{s}\}$:

$$\begin{aligned}
\mathcal{W}(x\,\texttt{.}) &= \{x\} \\
\mathcal{W}(x\,\texttt{:-}\,y\,\texttt{.}) &= \varnothing \\
\mathcal{W}(\mathcal{P}'x\,\texttt{.}) &= \mathcal{W}(\mathcal{P}') \cup \{x\} \\
\mathcal{W}(\mathcal{P}'x\,\texttt{:-}\,y\,\texttt{.}) &= \begin{cases} \mathcal{W}(\mathcal{P}') \cup \{x\} & \textit{falls } y \in \mathcal{W}(\mathcal{P}') \\ \mathcal{W}(\mathcal{P}') & \textit{sonst} \end{cases}
\end{aligned}$$

Für alle **SASP** Programme $\mathcal{P}$ gilt also $\mathcal{W}(\mathcal{P}) \subseteq \{\texttt{p}, \texttt{q}, \texttt{r}, \texttt{s}\}$.

Geben Sie für die folgenden drei Ausdrücke an, ob es sich um ein syntaktisch korrektes **SASP** Programm handelt und welche Semantik es hat.

i) `p. q :- p` ii) `p. q :- p. s :- r.` iii) `a :- r.`

b) Beweisen oder widerlegen Sie, dass im Allgemeinen gilt: Zwei Ausdrücke mit gleicher Semantik haben auch die gleiche Syntax.

c) Beweisen oder widerlegen Sie, dass im Allgemeinen gilt: Ein syntaktisch korrektes Programm ist auch semantisch korrekt.

Lösung:

a) i) Das Programm ist syntaktisch nicht korrekt (der `.` fehlt am Ende) und hat daher keine Semantik.

ii) Das Programm ist syntaktisch korrekt und seine Semantik ist $\{\texttt{p}, \texttt{q}\}$.

iii) Das Programm ist syntaktisch nicht korrekt (`a` ist kein Terminalsymbol der Grammatik G_2) und hat daher keine Semantik.

b) Die Aussage ist falsch. Betrachten wir die beiden syntaktisch korrekten, aber verschiedenen **SASP** Programme `p. q.` und das Programm ii) aus der vorigen Teilaufgabe. Beide haben die gleiche Semantik $\{\texttt{p}, \texttt{q}\}$, sind jedoch syntaktisch verschieden. Dieses Gegenbeispiel widerlegt die Aussage.

c) Die Aussage ist falsch. Ein Programm ist semantisch korrekt, wenn es genau die Anforderungen erfüllt, für die es entwickelt wurde. Lautet die Anforderung, dass das Programm die ersten eintausend Primzahlen ausgeben soll, so ist ein syntaktisch korrektes Programm, welches stattdessen "`Hello world!`" ausgibt, nicht semantisch korrekt. Dieses Gegenbeispiel widerlegt die Aussage.

Tutoraufgabe 3 (Formale Sprachen und Grammatiken):

Gegeben sei die folgende Sprache:

$$L_1 = \{w \in \{a, b\}^* \mid \text{ Auf ein } a \text{ folgt nie ein } b \text{ oder auf ein } b \text{ folgt nie ein } a\}$$

Die folgenden Wörter sind beispielsweise in der Sprache enthalten:

$$aaab \qquad bbaa \qquad aa \qquad \varepsilon$$

Folgende Wörter sind nicht Bestandteil der Sprache:

$$bab \qquad abba \qquad baba$$

a) Geben Sie eine kontextfreie Grammatik an, welche die Sprache L_1 erzeugt.

b) Geben Sie eine Grammatik in EBNF an, die L_1 definiert. Ihre Grammatik darf nur aus einer Regel bestehen und diese Regel darf nicht rekursiv sein (d. h. das Nichtterminalsymbol auf der linken Seite darf rechts nicht auftreten).

Um die Lesbarkeit zu erhöhen, dürfen Sie Anführungszeichen um Terminalsymbole weglassen.

c) Geben Sie ein Syntaxdiagramm ohne Nichtterminalsymbole an, das die Sprache L_1 definiert.

Lösung:

a) Die kontextfreie Grammatik $G_3 = (\{S_3, F, H\}, \{a, b\}, P_3, S_3)$ erzeugt genau die Sprache L_1, wobei P_3 genau die folgenden Produktionen enthält:

$$\begin{aligned} S_3 &\to FH \\ S_3 &\to HF \\ F &\to \varepsilon \\ F &\to aF \\ H &\to \varepsilon \\ H &\to bH \end{aligned}$$

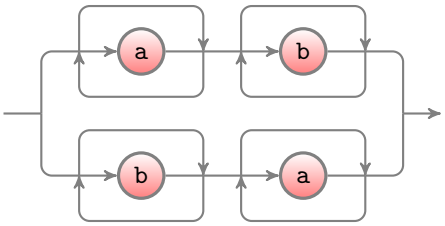
Mit dem Nonterminal S_3 wählt man, ob man zuerst a oder b Symbole erzeugen will. Das Nonterminal F erzeugt Wörter bestehend aus beliebig vielen a Symbolen, während H Wörter aus beliebig vielen b Symbolen erzeugt.

b) Die folgende Grammatik in EBNF mit nur einer nicht-rekursiven Regel definiert genau L_1.

$$S_3 = (\underbrace{\{a\}\{b\}}_{1} \mid \underbrace{\{b\}\{a\}}_{2})$$

Diese Konstruktion ist analog zu der Grammatik aus Teilaufgabe a). In Teil (1) wird ein Wort erzeugt, bei dem beliebig viele a Symbole vor beliebig vielen b Symbolen stehen. In Teil (2) wird ein Wort erzeugt, bei dem beliebig viele b Symbole vor beliebig vielen a Symbolen stehen.

c) Das folgende Syntaxdiagramm definiert die Sprache L_1:



Aufgabe 4 (Formale Sprachen und Grammatiken): (2 + 2 + 2 = 6 Punkte)

Gegeben sei die folgende Sprache:

$$L_2 = \{w \in \{a, b, c\}^* \mid \text{ Auf jedes } b \text{ folgt (mindestens und unmittelbar) ein } a \text{ und } (\sharp_b(w) + \sharp_c(w)) \,\%\, 2 = 0\}$$

Hierbei bezeichnet $\sharp_b(w)$ bzw. $\sharp_c(w)$ die Anzahl der b bzw. c Terminalsymbole in dem Wort w.

Die folgenden Wörter sind beispielsweise in der Sprache enthalten:

$$baaba \qquad aaa \qquad baccc \qquad \varepsilon$$

Folgende Wörter sind nicht Bestandteil der Sprache:

$$bababa \qquad acb \qquad bba$$

a) Geben Sie eine kontextfreie Grammatik an, welche die Sprache L_2 erzeugt.

b) Geben Sie eine Grammatik in EBNF an, die L_2 definiert. Ihre Grammatik darf nur aus einer Regel bestehen und diese Regel darf nicht rekursiv sein (d. h. das Nichtterminalsymbol auf der linken Seite darf rechts nicht auftreten).

Um die Lesbarkeit zu erhöhen, dürfen Sie Anführungszeichen um Terminalsymbole weglassen.

c) Geben Sie ein Syntaxdiagramm ohne Nichtterminalsymbole an, das die Sprache L_2 definiert.

Lösung:

a) Die kontextfreie Grammatik $G_4 = (\{S_4, E, O\}, \{a, b, c\}, P_4, S_4)$ erzeugt genau die Sprache L_2, wobei P_4 genau die folgenden Produktionen enthält:

$$\begin{aligned} S_4 &\to E \\ E &\to \varepsilon \\ E &\to aE \\ E &\to baO \\ E &\to cO \\ O &\to aO \\ O &\to baE \\ O &\to cE \end{aligned}$$

Das Nonterminal E erzeugt ein Wort mit einer geraden Anzahl an Elementen der Menge $\{b, c\}$. O dagegen erzeugt ein Wort mit einer ungeraden Anzahl solcher Elemente. Da das Terminalsymbol b nur mit anschließendem a erscheint, ist sichergestellt, dass auf jedes b ein a folgt.

b) Die folgende Grammatik in EBNF mit nur einer nicht-rekursiven Regel definiert genau L_2.

$$S_4 = \{(\underbrace{(c \mid ba)\{a\}(c \mid ba)}_{1} \mid a)\}$$

In Teil (1) wird ein Wort erzeugt, das genau zwei Terminalsymbole von $\{b, c\}$ enthält. Hier wird auch sichergestellt, dass auf jedes b ein a folgt. Von den so erzeugten Wörtern zusammen mit dem Terminalsymbol a können beliebig viele aneinandergereiht werden.

c) Das folgende Syntaxdiagramm definiert die Sprache L_2: