Theoretische Informatik (WS05)

Prof. Dr. Raimund Seidel
Wei Ding
Lösungsvorschlag zu Aufgabenblatt 6

Aufgabe 1

1.a

$P = \{$
$S \to aSa$, (1)
$S \to bSb$, (2)
$S \to aAb$, (3)
$S \to bAa$, (4)
$A \to aA$, (5)
$A \to bA$, (6)
$A \to \epsilon(7)$
$\}$
Die Ableitungen:

- $S \Rightarrow_4 bAa \Rightarrow_5 baAa \Rightarrow_5 baaAa \Rightarrow_6 baabAa \Rightarrow_6 baabbAa \Rightarrow_7 baabba$
- $S \Rightarrow_1 aSa \Rightarrow_2 abSba \Rightarrow_1 abaSaba \Rightarrow_1 abaaSaaba \Rightarrow_4 abaabAaaaba \Rightarrow_7 abaabaaaba$
- $S \Rightarrow_2 bSb \Rightarrow_1 baSab \Rightarrow_6 babAab \Rightarrow_7 babab$

1.b

$L_0 = \{wcw^R | c \in \{a, b, \epsilon\}, w \in \{a, b\}^*\}$
$L(G) = \overline{L_0}$

Es bleibt zu zeigen, dass $w \in L(G) \iff \exists Ableitung\ S \Rightarrow^* w$
Also zeigen wir, dass
$w \in L(G) \Rightarrow \exists Ableitung\ S \Rightarrow^* w$ und
$w \notin L(G) \Rightarrow \nexists Ableitung\ S \Rightarrow^* w$

Für alle Wörter x, die aus S abgeleitet werden können gilt $|x| \geq 2$.
Falls $w = vcv^R \in L_0$ gilt $\forall i = 0..\lfloor n/2 \rfloor\ \ w_i = w_{n-i}$ und w müsste zuerst mit Regeln 1,2 erzeugt werden, also $S \Rightarrow^* vSv^R$, und von S aus muss das erzeugte String länger als 1 sein

Nun bleibt zu zeigen, dass wenn $w \notin L_0$ es eine Ableitung für w in G gibt.

Sei $i \in \{0, .. \lfloor n/2 \rfloor\}$ die erste Stelle in w, für die gilt, dass $w_i \neq w_{n-i}$
Wir können mit Regeln 1,2 nun den ersten Teil bis zur Stelle i erzeugen und mit Regeln 3-7 jedes beliebige Wort dazwischen einfügen, dass mehr als ein Zeichen besitzt.

Theoretische Informatik (WS05)
Prof. Dr. Raimund Seidel
Wei Ding
Lösungsvorschlag zu Aufgabenblatt 6

Aufgabe 2

Sei G :

$$
\begin{aligned}
\Sigma = \{ & \quad a, b\} \\
V = \{ & \quad S, T, B, D, X\} \\
P = \{ & \\
& \quad S \to TBX, \\
& \quad TB \to aTDB, \\
& \quad DB \to BBD, \\
& \quad DX \to X, \\
& \quad B \to b, \\
& \quad T \to \epsilon, \\
& \quad X \to \epsilon\}
\end{aligned}
$$

Informelle Erklärung:

S ist das Startsymbol, das durch ein T ersetzt wird. Links von T werden die a's erzeugt, rechts davon die b's. D ist unser Verdopplungsoperator, der mit jedem a erzeugt wird und erst wieder entfernt werden kann, wenn alle B's verdoppelt wurden. X ist unser Endsymbol, welches das Ende des Wor tes markiert.

Warum ist $L(G) = L$?

„$L(G) \supseteq L$“

Das kürzeste Wort ($n = 0$) ist $a^0b^1 = b$. Dieses Wort wird erreicht durch $S \to TBX \to \epsilon BX \to \epsilon bX \to \epsilon b\epsilon = b$

Für ($n \geq 1$) werden die Wörter wie folgt produziert:

1.) Führe als erstes die Regel $S \to TBX$ aus.

2.) Führe n-mal die Regel $TB \to aTDB$

3.) Nach jedem Ausführen der zweiten Regel führe die Regeln $DB \to BBD$ und $DX \to X$ solange aus, bis keine $D's$ mehr da sind.

4.) Führe die Regeln $B \to b, T \to \epsilon$ und $X \to \epsilon$ solange aus, bis nur noch $a's$ und $b's$ vorhanden sind.

Es muss lediglich gezeigt werden, dass nach Schritt 1.) und n-maligem Aus führen der Schritte 2.) und 3.) unseres Algorithmus' neben einem T und einem X genau n $a's$ und 2^n $B's$ erzeugt wurden (da jedes B ein b wird und T und X nach dem viertem Schritt verschwinden, ist der Rest trivial). Beweis per Induktion:

Induktionsanfang: $n = 0$: bereits gesehen.

Induktionsbehauptung: nach n-maligem Ausführen der Schritte 2.) und 3.) wurden $n\,a's, 1\,T, 2^n\,B's$ und $1\,X$ erzeugt.

Induktionsschritt: $n \to n + 1$:

Nach n-maligem Ausführen der Schritte 2.) und 3.) $n\,a's, 1\,T, 2^n\,B's$ und $1\,X$ erzeugt (Ind. Beh.). Da alle $B's$ rechts von T und alle $a's$ links davon erzeugt werden, liegen nun 2^n $B's$ rechts von T. W ir wenden nun Regel 2.) an und erhalten ein zusätzliches a links von T. Außerdem haben wir nun ein D rechts von T und links von unser en 2^n $B's$. Nach jedem Anwenden der Regel $DB \to BBD$ wird 1 B rechts von D durch 2 $B's$ links von D ersetzt. Nach 2^n malig em Anwenden der Regel wurden also alle $B's$ verdoppelt und auf die links Seite von D verschoben. Wir haben jetzt $2 \times 2^n = 2^{n+1}$ $B's$. Das D steht jetzt unmittelbar links vom X und kann daher mit der R egel $DX \to X$ entfernt werden. An T und X wurde nichts geä ndert, die Anzahl der $a's$ wurde um 1 erhöht und die Anzahl der $B's$ verdoppelt. Daher stimmt die Induktionsbehauptung auch für n+1.

„$L(G) \subseteq L$“

Die Regel $S \to TBX$ wird immer als erstes ausgeführt und kann danach nicht mehr angewendet werden, so dass das Einfügen einzelner $B's$ danach nicht mehr möglich ist. Danach werden mit $TB \to aTDB$ die $a's$ eingeführt, und zwar ausschließlich links von T. Die $B's$ werden ausschließlich rechts von T eingeführt. $b's$ können nur a us $B's$ entstehen. Da es sonst keine Regeln gibt, in denen $B's$ links v on T oder einem a eingefügt werden und es ausserdem keine weiteren a-erzeugenden Regeln gibt, ist schonmal die äussere Form gesichert (er st $a's$, dann nur noch $b's$). Jetzt müssen wir nur noch sicherstellen, dass die Anzahl der Buchstaben korrekt ist. Dazu stellen wir fest: das i st genau dann der Fall, wenn für jedes eingefügte a die Anzahl der $b's$ verdoppelt wird.

Ein a kann nur über die Regel $TB \to aTDB$ eingefügt werden . In diesem Schritt haben wir neben dem a auch unseren „Verdopplungs operator“ D eingefügt. Dieser wird jetzt durch die Regel $DB \to BBD$ nach links geschiftet und verdoppelt dabei jedes weitere B. Am Ende des Wortes stösst D auf X und hat dann jedes B verdoppelt. D as D kann jetzt (und erst

jetzt) mit $DX \rightarrow X$ gelöscht werde n. Wenn n $a's$ eingefügt wurden, dann wurden auch n $D's$ eingefügt . Diese müssen alle $B's$ komplett durchlaufen und verdoppeln sie dabe i. Das bedeutet, dass die $B's$ nach n eingefügten $a's$ genau n-mal v erdoppelt werden. Da vor dem ersten Einfügen eines $a's$ genau 1 B vo rhanden ist, ergeben sich genau $1 \times 2^n B's$ und das ist genau die geforderte Anzahl. Die „Aufräumaktionen" $T \rightarrow \epsilon$ und $X \rightarrow \epsilon$ verändern die Anzahl der $B's$ und $a's$ nicht. Da aus jedem B genau ein b gemacht wird, liegt das erzeugt Wort in L.

Aufgabe 3

3.a
Eine Grammatik erzeugt Terminale durch ihre Ableitungsschritte. Der EA konsumiert sie durch Übergänge. Da jede Produktion einer linkslinearen Gramatik höchstens ein Terminal erzeugt, können wir uns für eine Produktion $(X \rightarrow Ya)$ die *Aufleitung* $(Y \rightarrow^a X)$ definieren, die wir mit dem Automaten abarbeiten.

Sei die Grammatik G $= (\Sigma, V, S, P)$ gegeben.
Den NEA M $= (\Sigma, Q, s \in Q, F \subset Q, \Delta)$, der genau alle Wörter akzeptieren soll, die G produziert, konstruieren wir auf folgende Weise:
$Q = V \cup \{s'\}$
$s = s'$
$F = \{S\}$

Warum entspricht der Enzustand gerade S ? Weil es für das Startsymbol keine weiteren Aufleitungen mehr gibt.
Wir konstruieren nun Δ:
Für jede Produktion $(X \rightarrow \epsilon)$ fügen wir eine Übergangsregel (s', ϵ, X) hinzu.
Warum? Weil wenn man ein Nichtterminal auf ϵ produziert, hat man das finale Wort abgeleitet. Und dieses Wort soll nun vom NEA eingelesen werden. Da der EA aber nur einen Startzustand haben kann, muss man von diesem aus mit ϵ-Übergängen zu jedem solchen Nichtterminal gelangen können.
Für jede Produktion $(X \rightarrow Ya)$ fügen wir eine Übergangsregel (Y, a, X) zu Δ hinzu, die die der Produktion entsprechende Aufleitung $(Y \rightarrow^a X)$ simuliert.
So ermöglichen wir, dass ein Automat, der sich im Zustand Y befindet, beim Einlesen von a in den Zustand X wechselt.
Daraus ergibt sich die formale Definition von Δ:
$\Delta = \{(s', \epsilon, X) | (X \rightarrow \epsilon) \in P\} \cup \{(Y, a, X) | (X \rightarrow Ya) \in P; X, Y \in V, a \in \Sigma\}$
Somit akzeptiert M genau die Worte, die G erzeugt. (Hier kleines schwarz-ausgefülltes Rechteck ein-texen)

Theoretische Informatik (WS05)

Prof. Dr. Raimund Seidel
Wei Ding

Lösungsvorschlag zu Aufgabenblatt 6

3.b

Zu einer jeden linkslinearen Grammatik G = (Σ, V, S, P) gibt es eine rechtslineare Grammatik G' = (Σ, V, S, P')
mit P' = $\{(X \to aY)|(X \to Ya) \in P\}$, die nach Konstrukion die Sprache L' = $\{w^R|w \in L(G)\} = (L(G))^R$ akzeptiert. Da in der Vorlesung gezeigt wurde, dass Sprachen zu rechtslinearen Grammatiken regulär sind, ist L' als solche regulär. L(G) ist nun aber genau $\{w^R|w \in L'\} = (L')^R$. Aus der Vorlesung wissen wir, dass wenn L regulär ist, auch L^R regulär ist. Somit ist L(G) regulär. q.e.d (Hier kleines schwarz-ausgefülltes Rechteck ein-texen)