RWTH Aachen **SS 2017**
Lehrgebiet Theoretische Informatik Blatt 10
Rossmanith–Dreier–Hark–Kuinke 10.7.2017

Lösungsvorschlag zur Vorlesung Formale Sprachen, Automaten und Prozesse

Aufgabe T28

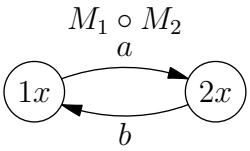
Wir zeigen, welche Fehler die Studenten gemacht haben:

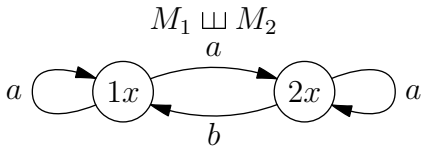
a)
Dieser Beweis ist falsch, da hier n konkret gewählt wurde. Wenn wir mit dem Pumping Lemma Regularität widerlegen, so müssen wir zeigen, dass es für jedes n ein Wort mit Länge größer oder gleich n in der Sprache gibt, welches der Spezifikation aus dem Pumping Lemma widerspricht. Für $n > 200$ findet man hier aber kein Wort mehr, da die Sprache endlich ist (jedes Wort hat höchstens Länge 200). Somit ist für $n = 201$ das Pumping Lemma trivialerweise erfüllt. L_2 ist endlich, also regulär.

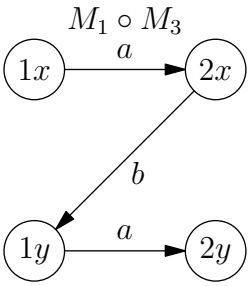
b)
Auch dieser Beweis ist falsch. Der Fehler liegt darin, dass die Zerlegung gewählt wurde, und es nicht für alle Zerlegungen durch Fallunterscheidung gezeigt wurde. Die gewünschte Aussage gilt nämlich nicht für alle Zerlegungen. Eine Zerlegung mit $x = \epsilon$, $y = abcd$ und $z = (abcd)^{n-1}$ könnte man zum Beispiel pumpen, ohne die Sprache zu verlassen. L_2 ist regulär, weil es durch den Regulären Ausdruck $(abcd)^*$ erkannt wird.

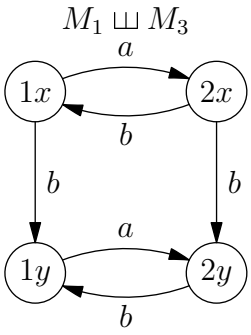
c)
Dieser Beweis ist aus mehreren Gründen falsch. Wieder wurde eine Zerlegung $uvwxy$, mit $|vwx| \leq n$ und $|vx| > 0$ gewählt. Man hätte jedoch alle Zerlegungen dieser Form betrachten müssen. Außerdem wurde angenommen, dass das Wort $a^m b^n a^n$ mit $m \neq n$ nicht in der Sprache ist. Tatsächlich ist jedoch jedes Wort $w \in \Sigma^*$ in der Sprache (mit $x = \epsilon$ und $y = w$). Daher ist die Sprache sogar regulär.

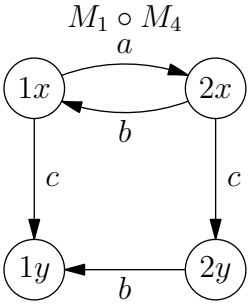
Aufgabe T29

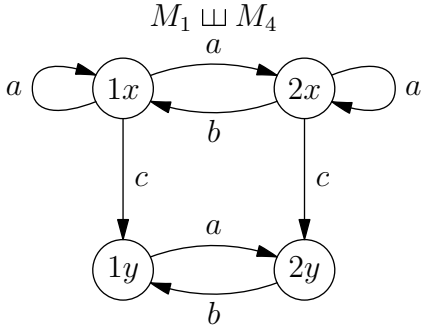












Aufgabe T30

In der Vorlesung haben wir eine Konstruktion kennen gelernt, die eine kontextfreie Grammatik in einen Kellerautomaten umwandelt. Dabei wurden die Ableitungsschritte durch ϵ-Transitionen implementiert, bei denen das oberste Symbol auf dem Keller durch seine Ableitung ersetzt wird.

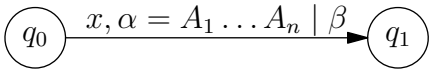
Bei einer kontextsensitiven Grammatik jedoch werden Nichtterminale in Abhängigkeit ihrer Umgebung ersetzt. Um dies durch einen Automaten zu simulieren, müsste dieser Zugriff auf mittlere Symbole des Stacks haben.

Man könnte sich also ein Automatenmodell überlegen, welches eine Liste, anstelle eines Stacks, als Speicher hat. Wie organisiert man den Zugriff auf diese Liste mit Hilfe der Übergangsrelation? Eine *Turingmaschine* ist ein mögliches solches Berechnungsmodell. Turingmaschinen sind sehr mächtig und erkennen genau die Chomsky-0 Sprachen. Mehr zu Turingmaschinen gibt es nächstes Semester in der Vorlesung "Berechenbarkeit und Komplexität".

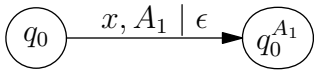
Aufgabe H24 (10 Punkte)

Wir zeigen, dass es eine solche Sprache nicht geben kann, da die Tiefgeragen- und Kellerautomaten äquivalent sind. Offensichtlich ist jeder Kellerautomat auch ein Tiefgaragenautomat. Nach Vorlesung existiert daher für jede kontextfreie Sprache auch ein Tiefgaragenautomaten, der sie erkennt. Wir skizzieren nun, wie aus einem Tiefgaragenautomat ein äquivalenter Kellerautomat konstruiert werden kann. Dabei müssen wir nur die neuen

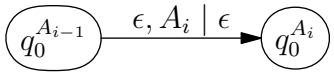
Transitionen verändern. Angenommen unser Tiefgaragenautomat enthält folgende Transition.



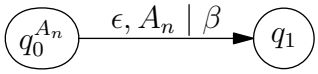
A_i sind Kellersymbole. Solch eine Transition zerlegen wir in eine Kette von Transitionen, wobei jedes Symbol A_i einzeln vom Keller eingelesen wird. Dabei fügen wir neue Zustände ein. Wir beginnen mit



Dann fügen wir für $2 \leq i \leq n-1$ die Transitionen



hinzu. Abschließend wird folgende Transition beigefügt.



Offenbar entspricht die Kette dieser Transitionen der Ausgangstransition und jede hinzugefügte Transition genügt der Definition eines Kellerautomaten. Wiederholen wir diese Konstruktion für jede Transition des Tiefgaragenautomaten, so erhalten wir einen Kellerautomaten, der dieselbe Sprache erkennt.

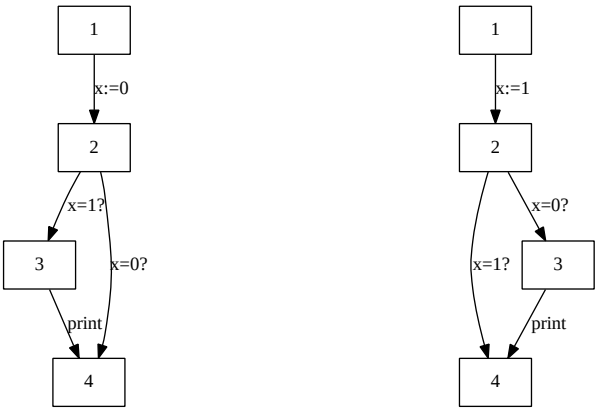
Aufgabe H25 (10 Punkte)

$$S \to abc \mid aSBc$$
$$cB \to Bc$$
$$bB \to bb$$

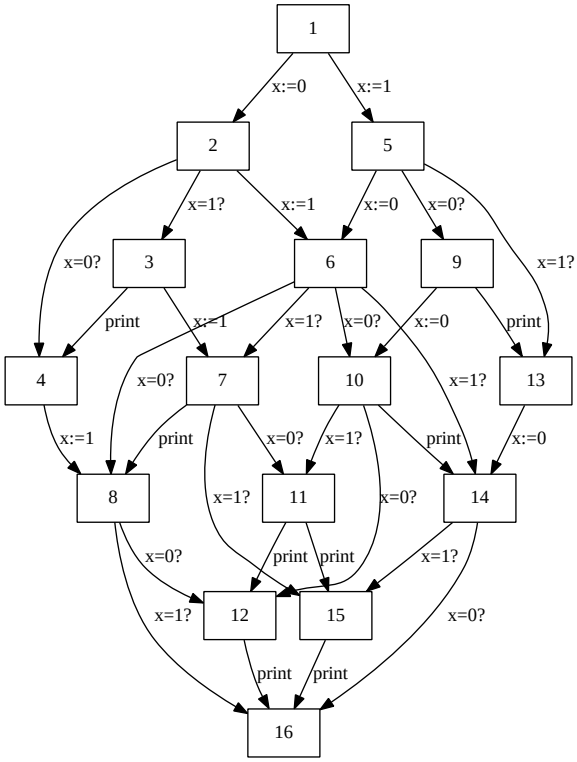
Es sei im folgenden kurz skizziert, warum diese Grammatik korrekt und vollständig ist. Wähle $n \geq 1$ beliebig. Dann gilt $S \to^* a^n bc(Bc)^{n-1}$. Mit der mittleren Regel erhalten wir durch mehrfaches Anwenden $a^n bc(Bc)^{n-1} \to^* a^n bB^{n-1}c^n$. Die letzte Regel führt zur Ableitung $a^n bB^{n-1}c^n \to^* a^n b^n c^n$. Dass abc in der Sprache der Grammatik ist, ist klar. Umgekehrt sieht man an den Produktionen der Grammatik, dass jede Ableitung von S zu einem Terminalwort w nach endlich vielen Schritten die Form $a^n b\beta c$ hat, wobei β aus $n-1$ Bs und cs in beliebiger Reihenfolge besteht. Nun kann ein B nur zu einem b abgeleitet werden, wenn es rechts neben einem b steht. Also muss w die Form $a^n b^n c^n$ für irgendein $n \geq 1$ haben.

Aufgabe H26 (15 Punkte)

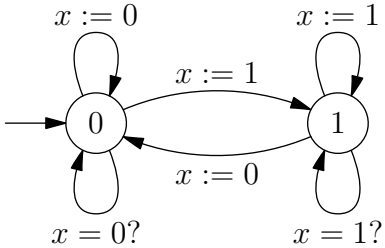
Wir beginnen mit der Modellierung der beiden Ablaufstrukturen P_1 und P_2 der Programme und erhalten:



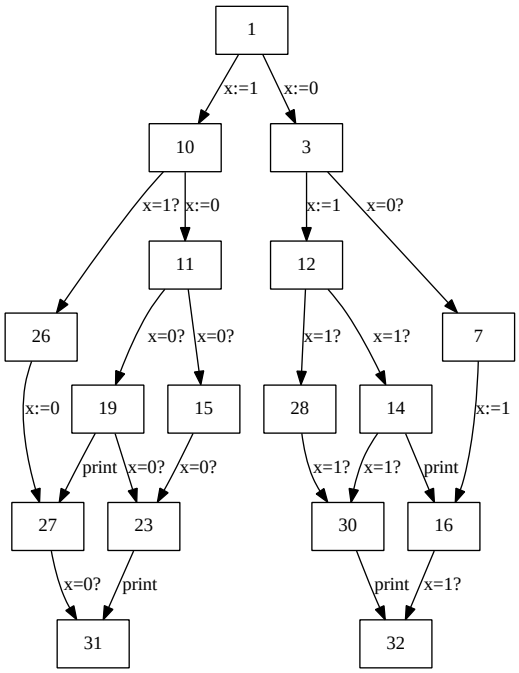
Das unsynchronisierte Produkt P_{12} von ihnen sieht dann so aus:



Den Automaten P_x für eine Variable x haben wir schon in der Vorlesung gesehen:



Schließlich berechnen wir $P_x \circ P_{12}$, welches ein Automat ist, dessen Sprache gerade alle möglichen Abläufe im Gesamtsystem enthält:



Wie zu erwarten fallen viele Läufe, welche in P_{12} noch möglich waren, jetzt weg. Insbesondere ist jetzt leicht zu erkennen, daß die Aktion `print` nicht zweimal ausgeführt werden kann.