Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 27. Oktober 2017 [1]

1. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Formale Sprachen 1+1+1 Punkte

Sei Σ ein beliebiges Alphabet und seien $L, L' \subseteq \Sigma^*$ Sprachen, welche aus endlich vielen Wörtern bestehen (d.h. $|L| \in \mathbb{N}$ und $|L'| \in \mathbb{N}$).
Beweisen oder widerlegen Sie folgende Aussagen:

(a) $L \cdot L' = L' \cdot L$

(b) $|L^n| = |L|^n$ für $n \in \mathbb{N}$

(c) $|\Sigma^n| = |\Sigma|^n$ für $n \in \mathbb{N}$

Aufgabe 2: Kleene-Abschluss 0,5+1,5+2 Punkte

Sei L eine Sprache, sodass $\varepsilon \in L$. Zeigen Sie, dass L^* die kleinste Sprache ist, die L enthält und unter Konkatenation abgeschlossen ist. Hierzu ist Folgendes zu zeigen:

(a) $L \subseteq L^*$

(b) $L^* \cdot L^* \subseteq L^*$

(c) Für eine Sprache L' mit $L \subseteq L'$ und $L' \cdot L' \subseteq L'$ gilt auch $L^* \subseteq L'$.

[1] Falls Sie ein Tutorat am Donnerstag oder Freitag besuchen, wird ausnahmsweise die Abgabefrist auf Montag, den 30. Oktober, um 14 Uhr verlängert, sodass Sie ausreichend Zeit zur Bearbeitung nach dem Besuch der Anwesenheitsübung haben.

Aufgabe 3: Endliche Automaten 4+2 Punkte

(a) Konstruieren Sie für die folgenden Sprachen L_i, $i \in \{1, 2, 3\}$, über dem Alphabet $\Sigma = \{a, b, c\}$ jeweils einen deterministischen endlichen Automaten (DEA) $\mathcal{A}_i$ mit $L_i = L(\mathcal{A}_i)$.

(i) $L_1 = \emptyset$

(ii) $L_2 = \{\varepsilon\}$

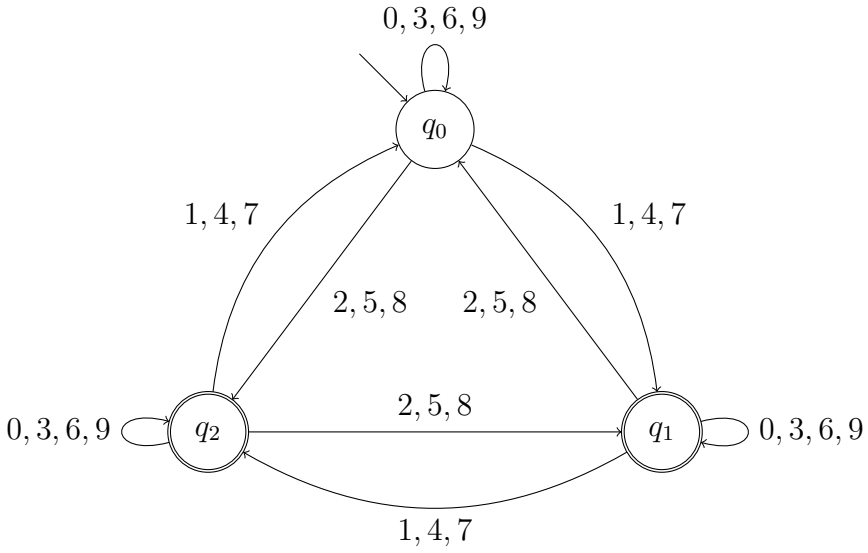
(iii) $L_3 = \{ubabv \mid u, v \in \Sigma^*\}$

Die graphische Darstellung der Automaten (Zustandsdiagramm) genügt.

(b) Betrachten Sie den folgenden deterministischen endlichen Automaten, welcher über dem Alphabet

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

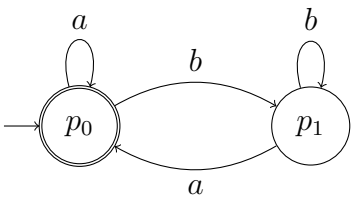
definiert ist. Welche Sprache wird von diesem Automaten erkannt? Geben Sie eine möglichst einfache Beschreibung dieser Sprache an.

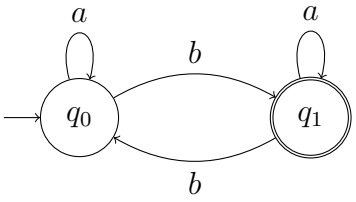


Hinweis: Interpretieren Sie ein Wort über dem Alphabet als Dezimaldarstellung einer natürlichen Zahl.

Aufgabe 4: Produktkonstruktion 2 Punkte

Betrachten Sie die folgenden beiden DEAs über dem Alphabet $\Sigma = \{a, b\}$.





Konstruieren Sie den Produktautomaten für Schnitt.

Hinweis: Überprüfen Sie zum Schluss (intuitiv), dass die Sprache des Produktautomaten gerade dem Schnitt der beiden anderen Sprachen entspricht.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 3. November 2017

2. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Rechtskongruenz I 3 Punkte

Betrachten Sie die Äquivalenzrelation $R_{\mathcal{A}}$ über beliebigem Alphabet Σ (s. Skript Bsp. 2.8):

$$R_{\mathcal{A}} = \{(u, v) \mid \tilde{\delta}(q^{\text{init}}, u) = \tilde{\delta}(q^{\text{init}}, v)\} \subseteq \Sigma^* \times \Sigma^*$$

Zeigen Sie, dass $R_{\mathcal{A}}$ rechtskongruent ist.

Hinweis: Sie dürfen die folgende Aussage ohne Beweis verwenden.
Für einen beliebigen DEA $(\Sigma, Q, \delta, q^{\text{init}}, F)$ gilt für alle $q \in Q$ und alle $u, v \in \Sigma^*$:

$$\tilde{\delta}(q, u \cdot v) = \tilde{\delta}(\tilde{\delta}(q, u), v).$$

Aufgabe 2: Rechtskongruenz II 3 Punkte

Betrachten Sie die folgende Relation $R \subseteq \Sigma^* \times \Sigma^*$ über $\Sigma = \{a, b\}$:

$$R = \{(u, v) \mid u \text{ enthält höchstens fünf } a \text{ genau dann, wenn } v \text{ höchstens fünf } a \text{ enthält}\}$$

(a) Ist R eine Äquivalenzrelation?

(b) Ist R rechtskongruent?

Begründen Sie Ihre Behauptungen.

Aufgabe 3: Nerode-Relation 5 Punkte

Betrachten Sie die Nerode-Relation R_L für die folgenden Sprachen über $\Sigma = \{a, b\}$.

$$L_1 = \{w \in \Sigma^* \mid w \text{ beginnt und endet mit einem } a\}$$

$$\begin{aligned} L_2 &= \{w \in \Sigma^* \mid w \text{ ist ein Palindrom}^1\} \\ &= \{w_0 w_1 \cdots w_n \in \Sigma^* \mid \text{für alle } i = 0, \ldots, n \text{ gilt } w_i = w_{n-i}\} \end{aligned}$$

(a) Geben Sie alle Äquivalenzklassen von R_{L_1} an. Begründen Sie, warum es keine weiteren Äquivalenzklassen gibt.

Konstruieren Sie anschließend den DEA aus Satz 2.5 (Myhill und Nerode), dessen Zustände gerade den Äquivalenzklassen entsprechen.

Hinweis: Sie können zur Hilfestellung zunächst einen DEA für L_1 konstruieren.

(b) Zeigen Sie, dass R_{L_2} einen unendlichen Index besitzt.

[1] Ein Wort w ist ein Palindrom, wenn w von rechts nach links gelesen wieder w ergibt.

Aufgabe 4: Markierungsalgorithmus 4 Punkte

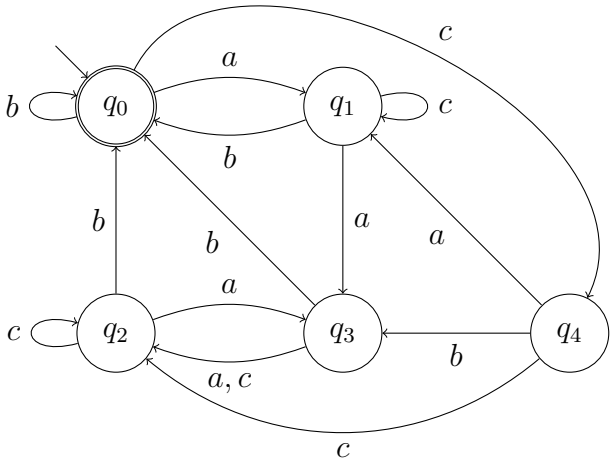
Markierungsalgorithmus

Eingabe: DEA $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$.

Ausgabe: Minimaler DEA für die Sprache $L(\mathcal{A})$.

1. Eliminiere in $\mathcal{A}$ alle nicht-erreichbaren Zustände.
2. Erstelle eine Tabelle, in der es für jedes Zustandspaar $\{q, q'\}$ mit $q \neq q'$ ein Feld gibt.
3. Markiere jedes Zustandspaar $\{q, q'\}$, für das $q \in F$ und $q' \notin F$ gilt.
4. Betrachte für jedes unmarkierte Zustandspaar $\{q, q'\}$ und jedes Symbol des Alphabets a das Zustandspaar $\{\delta(q, a), \delta(q', a)\}$. Ist $\{\delta(q, a), \delta(q', a)\}$ markiert, so markiere auch $\{q, q'\}$.
5. Wiederhole Schritt 4 so lange, bis es in der Tabelle keine Änderungen mehr gibt.
6. Fasse alle Zustände zusammen, deren Zustandspaare nicht markiert sind.

Wenden Sie den Markierungsalgorithmus auf den folgenden DEA über $\Sigma = \{a, b, c\}$ an. Geben Sie zusätzlich zum Ergebnisautomaten auch die verwendete Markierungstabelle an.



Hinweis: Da die Ordnung der Paare keine Rolle spielt, können Sie eine Hälfte der Tabelle ignorieren.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 10. November 2017

3. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: k-tes Symbol 2 Punkte

Betrachten Sie die folgende in $k > 0$ parametrisierte Sprache über $\Sigma = \{a, b\}$.

$$L_k = \{w \in \Sigma^* \mid \text{das } k\text{-te Symbol in } w \text{ ist ein } a\}$$

Geben Sie einen DEA $\mathcal{A}_k$ für ein beliebiges, gegebenes k an, der L_k akzeptiert.
Stellen Sie $\mathcal{A}_k$ als Struktur dar; ein Zustandsdiagramm genügt nicht.

Aufgabe 2: Minimale Anzahl der Zustände 3 Punkte

Aus der Vorlesung kennen Sie bereits die in $n > 0$ parametrisierte Sprache über $\Sigma = \{0, 1\}$

$$L_n = \{w \in \Sigma^* \mid \text{das } n\text{-letzte Zeichen von } w \text{ ist eine } 1\}.$$

Zeigen Sie, dass jeder DEA, der L_n akzeptiert, mindestens 2^n Zustände hat.

Aufgabe 3: Pumping Lemma I 2 Punkte

Das Pumping Lemma hat die Form einer Implikation $A \Rightarrow B$. Wir wenden es aber typischerweise in der umgekehrten (aber äquivalenten) Variante $\neg B \Rightarrow \neg A$ an.

Formulieren Sie das Pumping Lemma in der Form $\neg B \Rightarrow \neg A$, indem Sie in den Ausdrücken $\neg B$ und $\neg A$ die Negationen nach innen ziehen. Das heißt, dass keine Negation vor einem Quantor oder einem "und" bzw. "oder" stehen darf.

Hinweis: Eine kompakte Form des Pumping Lemmas:

$$\begin{aligned} &L \text{ ist regulär} \Rightarrow \\ &\quad (\exists n \in \mathbb{N}.\ n > 0 \wedge \forall z \in L.\ |z| \geq n \Rightarrow \exists u, v, w \in \Sigma^*.\ z = uvw \wedge |uv| \leq n \wedge |v| \geq 1 \\ &\qquad\qquad \wedge \forall i \in \mathbb{N}.\ uv^i w \in L) \end{aligned}$$

Aufgabe 4: Pumping Lemma II 4 Punkte

Zeigen Sie mithilfe des Pumping Lemmas, dass die folgende Sprache über $\Sigma = \{a, b\}$ nicht regulär ist.

$$L = \{a^m b^n \mid m < n\}$$

Aufgabe 5: NEA 2 Punkte

Geben Sie einen NEA an, der die folgende Sprache über $\Sigma = \{a, b\}$ akzeptiert.

$$L = \{w \in \Sigma^* \mid \exists u, v \in \Sigma^*.\ w = uabav\}$$

Hinweis: Ein NEA akzeptiert eine Sprache L, wenn gilt:

$$L = \{w \in \Sigma^* \mid \text{es gibt einen initialen, akzeptierenden Lauf über } w\}$$

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 17. November 2017

4. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Pumping Lemma 4 Punkte

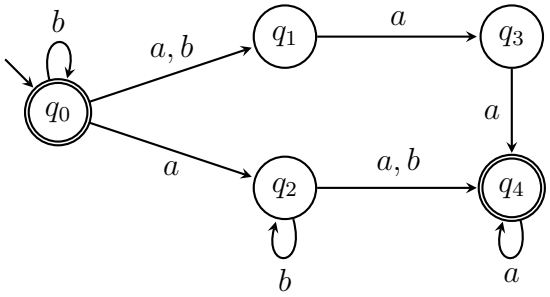
Betrachten Sie die Sprache der unär codierten Quadratzahlen über $\Sigma = \{\mathbf{1}\}$.

$$L = \left\{\mathbf{1}^k \mid k \text{ ist eine Quadratzahl}\right\}$$

Zeigen Sie mit dem Pumping Lemma, dass L nicht regulär ist.

Aufgabe 2: Potenzmengenkonstruktion 3 Punkte

Betrachten Sie den folgenden NEA über dem Alphabet $\Sigma = \{a, b\}$.



Konstruieren Sie einen DEA, der dieselbe Sprache akzeptiert. Verwenden Sie dazu die in der Vorlesung vorgestellte Potenzmengenkonstruktion. Dabei dürfen Sie nicht-erreichbare Zustände weglassen.

Aufgabe 3: ε-NEA 2 Punkte

Betrachten Sie das Alphabet $\Sigma = \{a, b\}$. Geben Sie einen ε-NEA an, der die Sprache

$$\{w \in \Sigma^* \mid \#_a(w) = 2 \text{ oder } \#_b(w) = 3\}$$

akzeptiert. Hierbei bezeichnet $\#_z(w)$ für alle $z \in \Sigma$ und $w \in \Sigma^*$ die Anzahl der Zeichen z, die in w vorkommen.

Aufgabe 4: Beweis zur ε-Eliminierung 2 Punkte

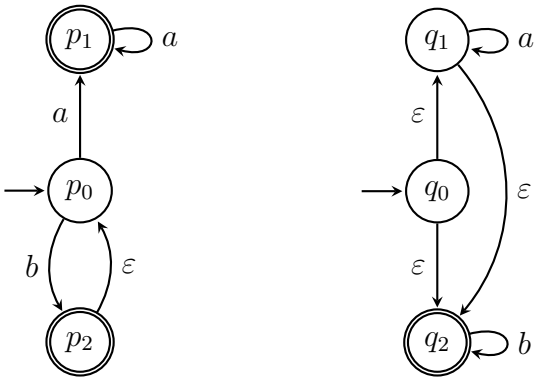
Satz 2.8 aus der Vorlesung besagt, dass es zu jedem ε-NEA einen NEA gibt, der die gleiche Sprache akzeptiert. Im Beweis dieses Satzes haben wir für einen ε-NEA $\mathcal{B}$ einen ε-freien NEA $\mathcal{N}$ konstruiert und die folgende Aussage verwendet.

$$\forall w \in \Sigma^+ \; \forall q, q' \in Q :$$
$$(q, w, q') \in \mathsf{reach}_{\mathcal{B}} \;\Leftrightarrow\; \exists \underbrace{q_0, q_1, \ldots, q_n}_{\text{Lauf von } \mathcal{N}} \in Q, \text{ sodass } n = |w|, q_0 = q \text{ und } q_n = q'$$

Zeigen Sie, dass diese Aussage gilt.

Aufgabe 5: Konkatenation und ε-Eliminierung 1+3 Punkte

Betrachten Sie die folgenden ε-NEAs $\mathcal{B}_1$ und $\mathcal{B}_2$ über dem Alphabet $\Sigma = \{a, b\}$:



(a) Konstruieren Sie einen ε-NEA $\mathcal{B}_3$, der die Sprache $L(\mathcal{B}_1) \cdot L(\mathcal{B}_2)$ akzeptiert. Verwenden Sie dazu die Konstruktion aus der Vorlesung (Definition 2.19).

(b) Eliminieren Sie anschließend die ε-Kanten aus $\mathcal{B}_3$. Verwenden Sie wieder die Konstruktion aus der Vorlesung (Definition 2.18).

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 24. November 2017

5. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Reguläre Ausdrücke 2,5 Punkte

Geben Sie reguläre Ausdrücke an, welche die folgenden Sprachen über dem Alphabet $\Sigma = \{a, b\}$ beschreiben.

(a) $L_1 = \{w \in \Sigma^* \mid$ auf jedes a in w folgt direkt ein b $\}$

(b) $L_2 = \{w \in \Sigma^* \mid$ w enthält das Teilwort $bb\}$

(c) $L_3 = \{w \in \Sigma^* \mid$ w enthält das Teilwort bb nicht$\}$

(d) $L_4 = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ enthält genau zweimal das Symbol } a \text{ oder } w \text{ enthält} \\ \text{genau einmal das Symbol } b \end{array} \right\}$

(e) Die Sprache der Wörter mit einer geraden Anzahl b's am Ende:

$L_5 = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} \text{die Länge des längsten Suffixes von } w, \text{ welches nur aus} \\ b\text{'s besteht, ist gerade} \end{array} \right\}$

Aufgabe 2: Regulärer Ausdruck $\rightsquigarrow$ endlicher Automat 2 Punkte

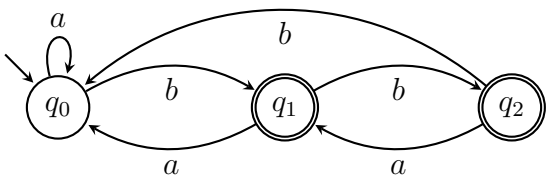
Betrachten Sie den folgenden regulären Ausdruck über $\Sigma = \{a, b\}$.

$$(b \cdot (a \cdot b))^*$$

Konstruieren Sie zu diesem Ausdruck einen äquivalenten endlichen Automaten. Verwenden Sie jeweils die im Skript vorgestellten Konstruktionen (Satz 2.12 „$\Leftarrow$“). Zustände dürfen Sie zur Vereinfachung umbenennen.
Es genügt, den resultierenden Automaten anzugeben. Wie immer gilt: Je gründlicher sie die einzelnen Schritte dokumentieren, desto eher bekommen Sie im Falle eines Fehlers noch Teilpunkte.

Aufgabe 3: Endlicher Automat $\rightsquigarrow$ regulärer Ausdruck 3 Punkte

Betrachten Sie den folgenden DEA $\mathcal{A}$ über $\Sigma = \{a, b\}$.



Bestimmen Sie für $\mathcal{A}$ das Gleichungssystem analog zur Vorlesung (Skript vor Bsp. 2.23). Berechnen Sie anschließend einen äquivalenten regulären Ausdruck, indem Sie das Gleichungssystem nach r_0 auflösen (Beweis zu Satz 2.12 „⇒“).

Sie dürfen reguläre Ausdrücke α, β, γ folgendermaßen vereinfachen. Für die Operationen „Konkatenation“ und „Oder“ gelten die folgenden Regeln:

$$\begin{aligned}
\text{Assoziativität:} &\quad \alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma, \;\; \alpha(\beta\gamma) = (\alpha\beta)\gamma \\
\text{Kommutativität:} &\quad \alpha + \beta = \beta + \alpha \\
\text{Neutrale Elemente:} &\quad \emptyset + \alpha = \alpha, \;\; \varepsilon\alpha = \alpha, \;\; \alpha\varepsilon = \alpha \\
\text{Distributivität:} &\quad \alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma, \;\; (\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma \\
\text{Absorption:} &\quad \emptyset\alpha = \emptyset, \;\; \alpha\emptyset = \emptyset
\end{aligned}$$

Für den Sternoperator gelten die folgenden Regeln:

$$\varepsilon^* = \varepsilon, \;\; (\varepsilon + \alpha)^* = \alpha^*, \;\; (\varepsilon + \alpha)\alpha^* = \alpha^*, \;\; \alpha^*(\varepsilon + \alpha) = \alpha^*$$

Aufgabe 4: Ableitung in einer Grammatik 1 Punkt

Gegeben sei die folgende Grammatik $G = (\Sigma, N, P, S)$ mit der Menge der Terminalsymbole $\Sigma = \{a\}$, der Menge der Nichtterminalsymbole $N = \{S, T, A, B, C, D, E, F, G\}$, dem Startsymbol S und den folgenden Regeln in P:

$$(1)\; S \to BT$$

(2)	T	$\to$	$AT \mid DCE$	(5)	AD	$\to$	FD	(8)	CD	$\to$	GD	(11)	B	$\to$ a
(3)	E	$\to$	$DCE \mid a$	(6)	FD	$\to$	FCA	(9)	GD	$\to$	GC	(12)	A	$\to$ a
(4)	BD	$\to$	BC	(7)	FC	$\to$	DC	(10)	GC	$\to$	DC	(13)	C	$\to$ a

Dabei ist $X \to Y \mid Z$ eine Abkürzung für $X \to Y, X \to Z$.
Geben Sie eine Ableitung für das Wort *aaaa* in der Grammatik an (mit allen Zwischenschritten).

Aufgabe 5: Reguläre Ausdrücke 2 Punkte

Betrachten Sie das Alphabet $A = \{a_1, \ldots, a_n\}$. Geben Sie eine kontextfreie Grammatik an, die die Menge der (vollständig geklammerten) regulären Ausdrücke über A erzeugt. Benutzen Sie dazu die folgenden Terminalsymbole:

$$\Sigma = A \cup \left\{ \boxed{\emptyset}, \boxed{\varepsilon}, \boxed{+}, \boxed{\cdot}, \boxed{^*}, \boxed{(}, \boxed{)} \right\}$$

Aufgabe 6: grep 2,5 Punkte

Das Unix Kommandozeilentool `grep` dient dem Finden und Filtern von Zeichenketten. Das Tool is auf allen gängingen Linuxdistributionen vorinstalliert und Sie haben es in der Vorlesung bereits kurz gesehen. Für einen regulären Ausdrucks `REGEXP` betrachtet der Befehl

```
grep -Er "REGEXP" .
```

alle Dateien im aktuellen Verzeichnis und dessen Unterverzeichnissen und liefert jeweils alle Zeilen, die ein Wort enthalten, das in der Sprache von `REGEXP` liegt.
Ihre Aufgabe ist es nun, für jede der folgenden Teilaufgaben einen regulären Ausruck für `grep` anzugeben. Wir wollen dabei `grep` mit dem Argument `-E` verwenden (`E` für *Extended* Regular Expressions).
Die Syntax von `grep` ist sehr reichhaltig. Wir wollen uns deshalb auf die folgenden Konstrukte beschränken.

Einzelne ASCII-Zeichen	Entsprechen den Elementen des Alphabets. Dabei müssen Sonderzeichen typischerweise mit einem Backslash maskiert werden. Wir schreiben also z.B. „`\?`“ statt „`?`“.
Punkt „.“	Beschreibt ein beliebiges Zeichen.
Runde Klammern „(“, „)“	Entsprechen den runden Klammern in der Notation der Vorlesung.
Strich-Operator „|“	Entspricht dem Plus-Operator für reguläre Ausdrücke aus der Vorlesung.
Stern-Operator „*“	Entspricht dem Stern-Operator für reguläre Ausdrücke aus der Vorlesung.
Eckige Klammern „[“, „]“	Mit eckigen Klammern lässt sich eine *Zeichenauswahl* beschreiben. So steht z.B. `[A-Za-z0-9]` für einen beliebigen Buchstaben oder eine beliebige Ziffer.
Hutsymbol „^“	Zeilenanfang
Dollarsymbol „$“	Zeilenende

(a) Zeilen, die eine Freiburger Telefonnummer enthalten.

Eine Freiburger Telefonnummer beginnt mit `+49` oder `0`, anschließend kommt die Folge `761` und anschließend eine beliebig lange Folge von Ziffern.

(b) Zeilen, die einen *schönen Domainnamen* aus Fidschi enthalten.

Ein schöner Domainname besteht nur aus kleinen Buchstaben, Ziffern und Punkten. Dabei darf der Domainname nicht mit einem Punkt beginnen und es dürfen niemals zwei Punkte aufeinander folgen. Außerdem soll am Ende die Top-Level-Domain von Fidschi `.fj` stehen.

(c) Zeilen, die eine *schöne E-Mailadresse* aus Fidschi enthalten.

Eine schöne E-Mailadresse beginnt mit einer nicht leeren Folge, die aus kleinen Buchstaben, Ziffern und dem Minuszeichen besteht. Es folgt das At-Symbol (@) und ein schöner Domainname aus Fidschi.

(d) Zeilen, die höchstens Leerzeichen enthalten.

(e) Zeilen, die einen HTML-Tag ohne Attribut enthalten.

HTML-Tags beginnen und enden mit spitzen Klammern. Ein HTML-Tag ohne Attribut hat zwischen diesen Klammern genau eine nicht leere Folge von Buchstaben und Ziffern. Zwischen dieser Folge und den Klammern ist aber noch eine beliebig lange Folge von Leerzeichen erlaubt. Beispiele sind `<h1>`, `< title >`.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 1. Dezember 2017

6. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Ableitungsbaum 1 Punkt

Gegeben sei die kontextfreie Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ mit $\Sigma = \{a, b\}$, $N = \{S\}$ und

$$\begin{aligned} P = \{ & S \to \varepsilon, \\ & S \to aSbS, \\ & S \to bSaS\}. \end{aligned}$$

Geben Sie einen Ableitungsbaum für das Wort *abbbaa* an.

Aufgabe 2: Dangling Else 1 Punkt

Gegeben sei die kontextfreie Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ mit

$$\Sigma = \{\texttt{if}, \texttt{then}, \texttt{else}, \texttt{:=}, \texttt{+1}, \texttt{-1}, \texttt{x}, \texttt{y}, \texttt{=0}\}, N = \{\mathit{Prog}, \mathit{Cond}, \mathit{Var}\}, S = \mathit{Prog}$$

und den folgenden Regeln in P:

$$\begin{aligned} \mathit{Prog} &\to \texttt{if}\ \mathit{Cond}\ \texttt{then}\ \mathit{Prog} \\ \mathit{Prog} &\to \texttt{if}\ \mathit{Cond}\ \texttt{then}\ \mathit{Prog}\ \texttt{else}\ \mathit{Prog} \\ \mathit{Prog} &\to \mathit{Var}\ \texttt{:=}\ \mathit{Var}\ \texttt{+1} \\ \mathit{Prog} &\to \mathit{Var}\ \texttt{:=}\ \mathit{Var}\ \texttt{-1} \\ \mathit{Cond} &\to \mathit{Var}\ \texttt{=0} \\ \mathit{Var} &\to \texttt{x} \\ \mathit{Var} &\to \texttt{y} \end{aligned}$$

Zeigen Sie, dass $\mathcal{G}$ nicht eindeutig ist.

Aufgabe 3: Palindrome 1+4 Punkte

Die Sprache der *Palindrome* über dem zweibuchstabigen Alphabet $\Sigma = \{a, b\}$ ist wie folgt definiert.

$$L_{Pal} = \{w_0 w_1 \dots w_n \in \Sigma^* \mid \text{für alle } i = 0, \dots, n \text{ gilt } w_i = w_{n-i}\}$$

(a) Geben Sie eine kontextfreie Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ an, die L_{Pal} erzeugt und nur ein Nichtterminalsymbol enthält (d.h. $|N| = 1$).

(b) Beweisen Sie durch Induktion, dass $L(\mathcal{G}) = L_{Pal}$ gilt.

Aufgabe 4: (Co-)Erreichbarkeit 2 Punkte

Gegeben sei eine kontextfreie Grammatik $\mathcal{G} = (\Sigma, N, P, S)$. Wir definieren:

Definition 1 (Erreichbarkeit)**.**
Ein Symbol $X \in \Sigma \cup N$ heißt *erreichbar*, wenn $S \vdash^* w_1 X w_2$ mit $w_1, w_2 \in (\Sigma \cup N)^*$ gilt.

Definition 2 (Co-Erreichbarkeit)**.**
Ein Symbol $X \in \Sigma \cup N$ heißt *co-erreichbar*, wenn ein $w \in \Sigma^*$ existiert mit $X \vdash^* w$.

Definition 3 (überflüssige Regel)**.**
Eine Regel $p \in P$ heißt *überflüssig in* $\mathcal{G}$, wenn die Grammatik $\mathcal{G}' = (\Sigma, N, P \setminus \{p\}, S)$ die gleiche Sprache erzeugt, d.h. $L(\mathcal{G}) = L(\mathcal{G}')$ gilt.

Zeigen oder widerlegen Sie:
Sind für eine Regel $X \to v_1 \dots v_n \in P$ die Symbole $X, v_1, \dots, v_n$ sowohl erreichbar als auch co-erreichbar, so ist die Regel nicht überflüssig in $\mathcal{G}$.

Aufgabe 5: Separierte Grammatik 1 Punkt

Wenden Sie die Konstruktion „Sep“ aus der Vorlesung auf die folgende Grammatik an.
$\mathcal{G} = (\Sigma, N, P, S)$ mit $\Sigma = \{a, b\}$, $N = \{S\}$ und

$$P = \{\ S \to aSb \mid \varepsilon\ \}$$

Aufgabe 6: Längenbeschränkte Grammatik 1 Punkt

Wenden Sie die Konstruktion „Bin“ aus der Vorlesung auf die folgende Grammatik an.
$\mathcal{G} = (\Sigma, N, P, S)$ mit $\Sigma = \{a, b\}$, $N = \{S, A, B\}$ und

$$\begin{aligned} P = \{\ & S \to ASBS \mid \varepsilon, \\ & A \to a \mid SBS, \\ & B \to b\ \} \end{aligned}$$

Aufgabe 7: ε-freie Grammatik 2 Punkte

Betrachten Sie die Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ mit $\Sigma = \{a, b\}$, $N = \{S, A, B, C\}$ und

$$\begin{aligned} P = \{\ & S \to AS \mid AB \mid AA, \\ & A \to a \mid BB \mid C, \\ & B \to b \mid \varepsilon \\ & C \to a\ \} \end{aligned}$$

(a) Wenden Sie den Algorithmus aus dem Beweis von Satz 3.4 an, um die Menge Nullable($\mathcal{G}$) zu berechnen. Geben Sie hierbei die Menge `M` nach jeder Iteration der while-Schleife an.

(b) Wenden Sie die Konstruktion „Del“ aus der Vorlesung auf die Grammatik $\mathcal{G}$ an. Überspringen Sie dabei Schritt 2, d.h., wenden Sie nicht „Sep“ und „Bin“ an, da $\mathcal{G}$ bereits die entsprechende Form hat.

Aufgabe 8: Kettenregelfreie Grammatik 2 Punkte

Betrachten Sie die Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ mit $\Sigma = \{a, b\}$, $N = \{S, A, B, C\}$ und

$$\begin{aligned} P = \{\ & S \to A \mid B, \\ & A \to B \mid b, \\ & B \to A \mid C, \\ & C \to AB \mid a\ \} \end{aligned}$$

Wenden Sie die Konstruktion „UNIT“ aus der Vorlesung auf die Grammatik $\mathcal{G}$ an. Halten Sie sich dabei an folgende Anleitung.

(a) Geben Sie den Graphen für Schritt 1 explizit an.

(b) Solange Sie auf einen Zyklus treffen (Schritt 2), so wählen Sie zur Elimination (Schritt 3) dasjenige Nichtterminalsymbol, welches zuerst im deutschen Alphabet vorkommt (im Skript: A_1). Geben Sie anschließend die resultierenden Produktionsregeln und erneut den entsprechenden Graphen an.

(c) Geben Sie explizit eine topologische Sortierung (Schritt 4) an. Annotieren Sie beispielsweise die Knoten im Graphen mit Nummern wie im Skript.

(d) Geben Sie für Schritt 5 die Produktionsregeln nach jeder äußeren Schleifeniteration an.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 8. Dezember 2017

7. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: CYK-Algorithmus 3 Punkte

Betrachten Sie die kontextfreie Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ in Chomsky-Normalform mit $\Sigma = \{0, 1\}$, $N = \{S, A, B, C, D\}$ und den folgenden Produktionsregeln.

$$\begin{aligned} P = \{\ & S \to BB \mid CA \mid CD, \\ & A \to 0 \mid DC \mid CA, \\ & B \to CC \mid DB, \\ & C \to 1 \mid CD, \\ & D \to 0 \mid AA\ \} \end{aligned}$$

Wenden Sie den CYK-Algorithmus an, um das Wortproblem für die Grammatik $\mathcal{G}$ und das Wort $w = 010110$ zu entscheiden. Geben Sie die vollständige Tabelle an, die der Algorithmus produziert.[1] Liegt w in der Sprache von $\mathcal{G}$?

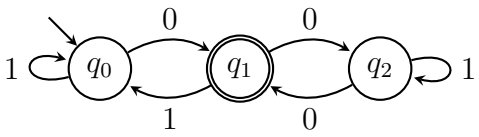
Aufgabe 2: DEA $\rightsquigarrow$ Typ-3-Grammatik 1+2+1 Punkte

Betrachten Sie die Konstruktion aus der „$\Rightarrow$“-Beweisrichtung von Satz 3.12 :

Sei $\mathcal{A} = (\Sigma, Q, \delta, q^{\text{init}}, F)$ ein DEA. Konstruiere die Typ-3-Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ mit

$$\begin{aligned} N &= Q \\ S &= q^{\text{init}} \\ P &= \{q \to aq' \mid q, q' \in Q, a \in \Sigma, \delta(q, a) = q'\} \cup \{q \to \varepsilon \mid q \in F\} \end{aligned}$$

(a) Betrachten Sie den folgenden DEA über dem Alphabet $\Sigma = \{0, 1\}$.



Geben Sie eine äquivalente Typ-3-Grammatik an. Verwenden Sie dazu die obige Konstruktion.

[1] In der Literatur gibt es zahlreiche Variationen des Algorithmus, die sich im Wesentlichen in der Anordnung der Tabelleneinträge M_{ij} unterscheiden. Verwenden Sie die Anordnung aus der Vorlesung. Der erste Index gibt die Zeilennummer an, der zweite Index gibt die Spaltennummer an. Die Zeilennummern werden von oben nach unten größer, die Spaltennummern werden von links nach rechts größer.

(b) Zeigen Sie das folgende Lemma zu obiger Konstruktion:

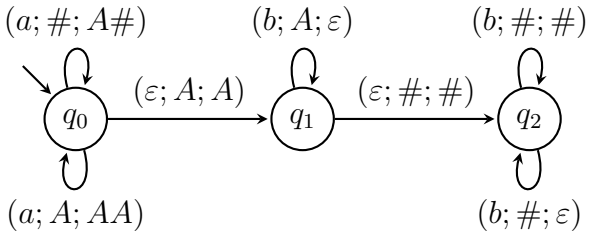
$$\delta^*(q^{\text{init}}, w) = q' \text{ gdw. } q^{\text{init}} \overset{*}{\vdash}_{\mathcal{G}} wq'$$

(c) Zeigen Sie mithilfe des Lemmas, dass $L(\mathcal{A}) = L(\mathcal{G})$ für obige Konstruktion gilt.

Anmerkung: Teilaufgaben (b) und (c) sind für einen beliebigen DEA zu zeigen, nicht nur für denjenigen aus dem Beispiel.

Aufgabe 3: Kellerautomaten I 2 Punkte

Betrachten Sie folgenden Kellerautomaten $\mathcal{K} = (\Sigma, Q, \Gamma, q^{\text{init}}, Z^{\text{init}}, \delta)$ mit dem Eingabealphabet $\Sigma = \{a, b\}$, der Menge von Zuständen $Q = \{q_0, q_1, q_2\}$, dem Kelleralphabet $\Gamma = \{\#, A\}$, dem Startzustand $q^{\text{init}} = q_0$ und dem Startsymbol des Kellers $Z^{\text{init}} = \#$. Im folgenden Zustandsdiagramm von $\mathcal{K}$ sind Beschriftungen $(x; Z; \gamma)$ von Transitionen für $x \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$ und $\gamma \in \Gamma^*$ folgendermaßen zu lesen: x ist das Eingabesymbol (oder ε) und Z ist das oberste Kellersymbol, welches nach Ausführung der Transition durch γ an der Spitze des Kellers ersetzt wird.



(a) Akzeptiert $\mathcal{K}$ das Wort *aabbb*?

(b) Was ist die von $\mathcal{K}$ erkannte Sprache $L(\mathcal{K})$?

Aufgabe 4: Kellerautomaten II 2 Punkte

Konstruieren Sie einen Kellerautomaten $\mathcal{K}$, der die folgende Sprache erkennt.

$$L = \{a^n b^{2n} \mid n \in \mathbb{N}\}$$

Geben Sie jede Komponente der Struktur $\mathcal{K} = (\Sigma, Q, \Gamma, q^{\text{init}}, Z^{\text{init}}, \delta)$ an. Sie dürfen die Transitionsfunktion δ durch ein Zustandsdiagramm wie in Aufgabe 3 darstellen.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 15. Dezember 2017

8. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Mehr Keller – mehr Möglichkeiten 3 Punkte

Sei $\mathcal{K} = (\Sigma, Q, \Gamma, q^{\text{init}}, Z^{\text{init}}, \delta)$ ein beliebiger Kellerautomat. Lemma 4.2 aus der Vorlesung besagt:

> Für alle $q, q' \in Q, w \in \Sigma^*, Z \in \Gamma$ und $n \in \mathbb{N}$ gilt:
>
> Wenn $(q, w, Z) \rhd^n (q', \varepsilon, \varepsilon)$, dann $\forall v \in \Sigma^*, \gamma \in \Gamma^* : (q, wv, Z\gamma) \rhd^n (q', v, \gamma)$.

Beweisen Sie dieses Lemma.

Hinweis: Es ist leichter, die folgende allgemeinere Behauptung zu zeigen.

> Für alle $q, q' \in Q, w \in \Sigma^*, \hat{\gamma} \in \Gamma^*$ und $n \in \mathbb{N}$ gilt:
>
> Wenn $(q, w, \hat{\gamma}) \rhd^n (q', \varepsilon, \varepsilon)$, dann $\forall v \in \Sigma^*, \gamma \in \Gamma^* : (q, wv, \hat{\gamma}\gamma) \rhd^n (q', v, \gamma)$.

Aufgabe 2: Mehr Keller – mehr Möglichkeiten „umgedreht“ 1 Punkt

Sei $\mathcal{K} = (\Sigma, Q, \Gamma, q^{\text{init}}, Z^{\text{init}}, \delta)$ ein beliebiger Kellerautomat. Wenn wir die Implikation in Lemma 4.2 umdrehen, lautet die Aussage:

> Für alle $q, q' \in Q, w \in \Sigma^*, Z \in \Gamma$ und $n \in \mathbb{N}$ gilt:
>
> Wenn $\forall v \in \Sigma^*, \gamma \in \Gamma^* : (q, wv, Z\gamma) \rhd^n (q', v, \gamma)$, dann $(q, w, Z) \rhd^n (q', \varepsilon, \varepsilon)$.

Gilt diese Aussage? Begründen Sie Ihre Antwort.

Aufgabe 3: CFG $\rightsquigarrow$ NPDA 2 Punkte

Betrachten Sie die kontextfreie Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ mit $\Sigma = \{a, b, c\}$, $N = \{S, A, B\}$ und den folgenden Regeln:

$$\begin{aligned} P = \{ & S \to Aa \mid Bc, \\ & A \to aaAb \mid a, \\ & B \to aBb \mid aA \mid \varepsilon \} \end{aligned}$$

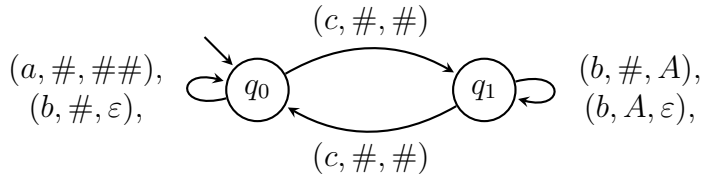
(a) Konstruieren Sie einen Kellerautomaten $\mathcal{K}$, sodass $L(\mathcal{K}) = L(\mathcal{G})$ gilt. Verwenden Sie dabei das in der Vorlesung vorgestellte Verfahren[1] (Lemma 4.1).

(b) Geben Sie eine akzeptierende Folge von Konfigurationen in Ihrem Kellerautomaten für das Wort *aaabc* an.

[1] Die Konstruktion wurde in der Vorlesung auf Grammatiken in CNF beschränkt, um den Beweis zu vereinfachen. Die Konstruktion funktioniert aber auch für allgemeine kontextfreie Grammatiken. Sie müssen die Grammatik daher nicht erst in CNF bringen.

Aufgabe 4: NPDA $\rightsquigarrow$ CFG 3 Punkte

Betrachten Sie den folgenden Kellerautomat $\mathcal{K} = (\Sigma, Q, \Gamma, q_0, \#, \delta)$ mit dem Eingabealphabet $\Sigma = \{a, b, c\}$, dem Kelleralphabet $\Gamma = \{\#, A\}$ und den restlichen Komponenten gegeben im folgenden Zustandsdiagramm.



(a) Geben Sie eine Grammatik $\mathcal{G}$ an, sodass $L(\mathcal{G}) = L(\mathcal{K})$ gilt. Verwenden Sie dabei das in der Vorlesung vorgestellte Verfahren (Lemma 4.4).

(b) Geben Sie eine Ableitung für das Wort *acbbcb* in Ihrer Grammatik an.

Aufgabe 5: Kellerautomaten mit akzeptierenden Zuständen 4 Punkte

Wir definieren einen *Kellerautomaten mit akzeptierenden Zuständen* als 7-Tupel

$$\mathcal{E} = (\Sigma, Q, \Gamma, q^{\text{init}}, Z^{\text{init}}, \delta, F)$$

mit den Komponenten $\Sigma, Q, \Gamma, q^{\text{init}}, Z^{\text{init}}, \delta$ wie für einen gewöhnlichen Kellerautomaten und $F \subseteq Q$ als die Menge der akzeptierenden Zustände.

Weiterhin definieren wir das Akzeptanzverhalten folgendermaßen:
Eine Konfiguration (q, w, γ) eines Kellerautomaten mit akzeptierenden Zuständen ist *akzeptierend*, falls $q \in F$ und $w = \varepsilon$. Die von einem Kellerautomaten mit akzeptierenden Zuständen $\mathcal{E}$ *akzeptierte Sprache* ist

$$L(\mathcal{E}) = \{w \in \Sigma^* \mid \exists q \in F, \gamma \in \Gamma^* : (q^{\text{init}}, w, Z^{\text{init}}) \rhd^* (q, \varepsilon, \gamma)\}.$$

Hierbei ist $\rhd$ die Schrittrelation genau wie für gewöhnliche Kellerautomaten.
Geben Sie für die folgenden Behauptungen jeweils eine Konstruktion (ohne Beweis) an:

(a) Für jeden gewöhnlichen Kellerautomaten $\mathcal{K}$ gibt es einen Kellerautomaten mit akzeptierenden Zuständen $\mathcal{E}$, welcher die gleiche Sprache akzeptiert.

(b) Für jeden Kellerautomaten mit akzeptierenden Zuständen $\mathcal{E}$ gibt es einen gewöhnlichen Kellerautomaten $\mathcal{K}$, welcher die gleiche Sprache akzeptiert.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 22. Dezember 2017

9. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Pumping Lemma für kontextfreie Sprachen 3 Punkte

Betrachten Sie die folgende Sprache über dem Alphabet $\Sigma = \{a, b\}^*$.

$$L = \{a^n b^{2n} a^n \mid n \in \mathbb{N}\}$$

Zeigen Sie mit dem Pumping Lemma für kontextfreie Sprachen, dass L nicht kontextfrei ist. Analoge Fälle müssen Sie nicht mehrfach betrachten; geben Sie die Fälle aber explizit an.

Aufgabe 2: Abschlusseigenschaften kontextfreier Sprachen 1+2 Punkte

(a) Betrachten Sie die beiden kontextfreien Grammatiken $\mathcal{G}_i = (\Sigma, N_i, P_i, S_i)$ für $i = 1, 2$ mit $\Sigma = \{a, b\}$, $N_i = \{S_i, A\}$ und den folgenden Produktionsregeln.

$$P_1 = \{\ S_1 \to A \mid \varepsilon$$
$$\qquad A \to a \mid AAb\ \}$$

$$P_2 = \{\ S_2 \to aA$$
$$\qquad A \to a \mid aAb\ \}$$

Konstruieren Sie eine kontextfreie Grammatik, welche die folgende Sprache erzeugt.

$$L(\mathcal{G}_1) \cdot (L(\mathcal{G}_2))^*$$

Verwenden Sie dazu die Konstruktionen aus der Vorlesung (Beweis zu Satz 5.5). *Hinweis:* Das Umbenennen von Nichtterminalsymbolen kann helfen um eine disjunkte Vereinigung zu ermöglichen.

(b) Seien L_1 und L_2 kontextfreie Sprachen. Ist die Differenz $L_1 \setminus L_2$ kontextfrei?

Beweisen Sie Ihre Behauptung.

Aufgabe 3: Deterministisch kontextfreie Sprachen 2 Punkte

Betrachten Sie die folgenden beiden Sprachen über dem Alphabet $\Sigma = \{a, b, c\}$.

$$L_1 = \{a^n b a^n \mid n \in \mathbb{N}\}$$
$$L_2 = \{a^n c a^{2n} \mid n \in \mathbb{N}\}$$

Zeigen Sie, dass die Vereinigung $L_1 \cup L_2$ deterministisch kontextfrei ist, indem Sie einen deterministischen Kellerautomaten angeben, der diese Sprache akzeptiert.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 12. Januar 2018

10. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Turingmaschinen, Funktion berechnen 2 Punkte

Betrachten Sie die Turingmaschine $\mathcal{M} = (\Sigma, Q, \Gamma, \delta, q^{\text{init}}, \sqcup, F)$ mit $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1\}$, $\Gamma = \Sigma \cup \{\sqcup\}$, $q^{\text{init}} = q_0$, $F = \emptyset$ und δ gegeben durch folgende Turingtafel.

q_0	0	q_0	1	R
q_0	1	q_0	0	R
q_0	$\sqcup$	q_1	$\sqcup$	L
q_1	0	q_1	0	N

Welche partielle Funktion f berechnet die Turingmaschine? Geben Sie sowohl eine umgangssprachliche als auch eine formale Beschreibung der partiellen Funktion f an.

Hinweis: Für undefinierte Ausgaben können Sie den Wert Ihrer partiellen Funktion f entweder ebenfalls undefiniert lassen oder explizit den Wert „undef.“ zurückliefern.

Aufgabe 2: Turingmaschinen, Sprache akzeptieren 2 Punkte

Betrachten Sie die Turingmaschine $\mathcal{M} = (\Sigma, Q, \Gamma, \delta, q^{\text{init}}, \sqcup, F)$ mit $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_\ell\}$, $\Gamma = \Sigma \cup \{\sqcup, *\}$, $q^{\text{init}} = q_0$, $F = \{q_1\}$ und δ gegeben durch folgende Turingtafel.

q_0	0	q_1	$*$	R
q_0	1	q_0	1	R
q_0	$*$	q_0	$*$	R
q_1	0	q_1	0	R
q_1	1	q_ℓ	$*$	L
q_1	$*$	q_1	$*$	R
q_ℓ	0	q_ℓ	0	L
q_ℓ	1	q_ℓ	1	L
q_ℓ	$*$	q_ℓ	$*$	L
q_ℓ	$\sqcup$	q_0	$\sqcup$	R

Welche Sprache akzeptiert die Turingmaschine? Geben Sie sowohl eine umgangssprachliche als auch eine formale Beschreibung der Sprache an.

Aufgabe 3: Jordi und die Lichterkette 3 Punkte

Nach vielen Jahren im Einsatz funktioniert die große Lichterkette am Nordpol nicht mehr. Der Weihnachtsmann hat den Elfen Jordi damit beauftragt, diese zu reparieren. Da die Kette zwar endlich, aber sehr lang sein kann und Jordi in der Vorweihnachtszeit sehr beschäftigt ist, hat er Sie um Hilfe gebeten. Mit einer verzauberten Turingmaschine, die als Eingabe die Lichterkette bekommt, sollen Sie alle Fehler beheben. Dabei startet die Maschine am linken Ende der Kette auf der ersten Lampe, ein Stück Kabel wird als $-$, eine funktionstüchtige Lampe als  und eine defekte Lampe als  dargestellt. Die möglichen Fehlerarten sind:

(a) Eine Lampe kann defekt sein, d.h. statt  steht  auf dem Band. Ersetzen Sie hier die defekte durch eine funktionierende Lampe.

(b) Ein Stück Kabel kann fehlen, d.h. statt $-$ steht $\sqcup$ auf dem Band. Wie das passieren konnte, weiß Jordi selbst nicht so genau, da das Zeichen $\sqcup$ gar nicht im Eingabealphabet vorkommt.[1] Trotzdem muss die Maschine fehlende Kabel ausbessern. Fügen Sie in diesem Fall ein Stück Kabel ein. Natürlich können auch mehrere Stücke des Kabels hintereinander fehlen. Es kann jedoch keine Lampe fehlen.

Zwei Lampen werden immer von ein bis zwei (möglicherweise defekten) Stücken Kabel getrennt. Die Kette startet und endet mit einer (möglicherweise defekten) Lampe. Formal sieht die Turingmaschine (ohne Zustände und Transitionen) folgendermaßen aus:

$$\tau = (\Sigma, Q, \Gamma, \delta, q_0, \sqcup, \emptyset) \text{ mit } \Sigma = \{-, \text{  }, \text{}\}, \Gamma = \Sigma \cup \{\sqcup\}, q_0 \in Q$$

Vervollständigen Sie die Maschine, indem Sie Zustände und Transitionen einfügen.

Hinweis: Beachten Sie, dass Sie kein zusätzliches Bandsymbol verwenden dürfen. Die Kette, die *am Ende* auf dem Band steht, darf nicht kürzer oder länger sein als die Eingabe. Die Turingmaschine muss auf allen gültigen Eingaben (d.h. eine möglicherweise defekte Lichterkette) terminieren. Testen Sie Ihre Turingmaschine dazu am besten mit einigen Beispieleingaben. Eine gültige Beispieleingabe und erwartete Ausgabe finden Sie unten.

Eingabe:

$\cdots$	$\sqcup$		$-$	$\sqcup$		$\sqcup$	$\sqcup$		$-$		$\sqcup$	$\cdots$

Ausgabe:

$\cdots$	$\sqcup$		$-$	$-$		$-$	$-$		$-$		$\sqcup$	$\cdots$

Frohe Feiertage und ein gutes neues Jahr wünschen

Marc Fuchs, Matthias Heizmann, Jacob Maxam, Jan Oreans,
Andreas Podelski, Saskia Rabald, Claus Schätzle und Christian Schilling.

[1] Nur bei dieser „verzauberten“ Turingmaschine kann ein Blank Teil der Eingabe sein. Bei normalen Turingmaschinen, wie wir sie in der Vorlesung kennengelernt haben, ist dies nicht erlaubt.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 19. Januar 2018

11. Übungsblatt zur Vorlesung
Informatik III

Anmerkung: Undokumentierte Turingmaschinen zu verstehen ist ähnlich schwierig wie undokumentierten Programmcode zu verstehen. Immer wenn Sie eine Turingmaschine mit einer nicht offensichtlichen Funktionalität angeben, sollten Sie auch in Textform beschreiben, wie diese Turingmaschine funktioniert.

Aufgabe 1: Flussdiagramme 4 Punkte

Betrachten Sie das Alphabet der Striche $\Sigma = \{I\}$. Wir interpretieren Wörter aus Σ^* als Unärzahlen.
Geben Sie für diese Aufgabe jeweils Turingmaschinen als Flussdiagramme an. Verwenden Sie dabei $\sqcup$ als Blank.

(a) Geben Sie eine Turingmaschine $\mathcal{M}_{++}$ an, die an eine beliebige Eingabe über dem Eingabealphabet $\{I, \#\}$ rechts I anhängt. Außerdem soll der Kopf zurück an die Ausgangsposition bewegt werden, sofern die Eingabe nicht leer war, und ansonsten auf das neu hinzugefügte I bewegt werden.

(b) Geben Sie eine Turingmaschine $\mathcal{M}_{\text{add}}$ an, die zwei Unärzahlen, die durch # getrennt sind, einliest und die erste Zahl zur zweiten Zahl addiert. Die erste Zahl und das Trennzeichen sollen dabei stehenbleiben.

(c) Geben Sie eine Turingmaschine $\mathcal{M}_{\text{mult}}$ an, die zwei positive Unärzahlen, die durch # getrennt sind, multipliziert. Die Ausgabe soll nur das Resultat der Multiplikation sein.

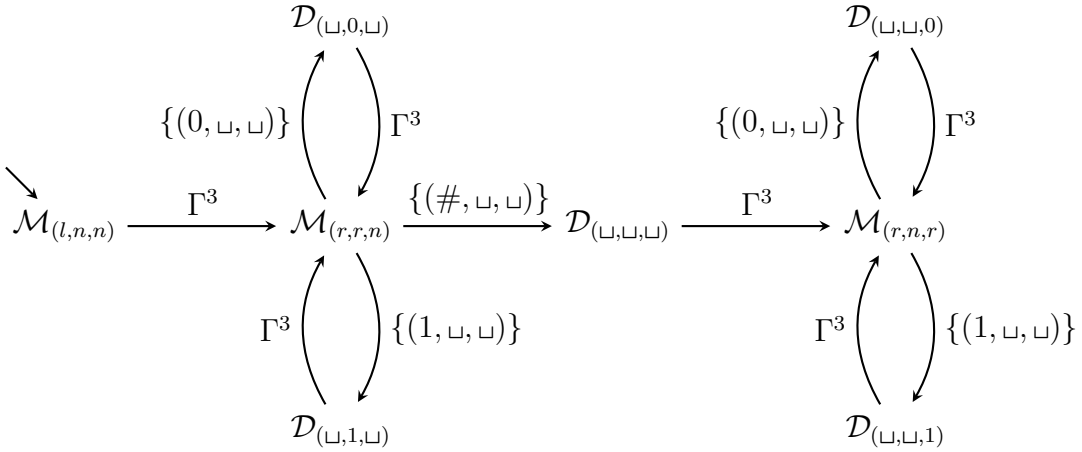
Hinweis: Die unäre Null wird mit dem leeren Wort dargestellt. Beispielsweise ist das Wort $w = \#$ eine gültige Eingabe für $\mathcal{M}_{\text{add}}$.

Aufgabe 2: Mehrband-Turingmaschine 3 Punkte

Geben Sie eine 3-Band-Turingmaschine an, welche die Binäraddition nach der Schulmethode durchführt. Zu Beginn sollen die Operanden getrennt durch # auf Band 1 stehen. Verwenden Sie die Konvention, dass Binärzahlen mit dem niederwertigsten Bit rechts geschrieben werden. Zum Beispiel steht die Binärzahl 10 für die Dezimalzahl 2.

Wir verallgemeinern hierfür die Flussdiagrammschreibweise auf k-Band-Turingmaschinen, indem wir an jede Kante statt einer Menge von Bandsymbolen eine Menge von k-Tupeln von Bandsymbolen schreiben.
Weiterhin führen wir die folgenden Verallgemeinerungen von Turingmaschinen aus der Vorlesung für drei Bänder ein: $\mathcal{D}_{(x_1,x_2,x_3)}$ ist die Druckmaschine, die auf Band i das Zeichen x_i schreibt. $\mathcal{M}_{(d_1,d_2,d_3)}$ ist die 1-Schritt-Maschine, die auf Band i einen Schritt in Richtung d_i macht, wobei l, r, n jeweils für „links“, „rechts“ oder „stehen bleiben“ steht.

Beispiel: Das folgende Flussdiagramm beschreibt eine Turingmaschine $\mathcal{M}_{\text{copy}}$, welche die Operanden auf die Bänder 2 und 3 verteilt, Band 1 leert und die Köpfe jeweils auf das erste Blank rechts von den Operanden bewegt.

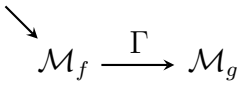


Ihre Lösung sollte $\mathcal{M}_{\text{copy}}$ verwenden.

Hinweis: Beachten Sie, dass die Operanden nicht die gleiche Länge haben müssen.

Aufgabe 3: Hintereinanderausführung von Funktionen 2 Punkte

Sei Σ ein beliebiges Alphabet. Seien $\mathcal{M}_f$ und $\mathcal{M}_g$ zwei deterministische TMs mit Bandalphabet Γ. Sei $f : \Sigma^* \to \Sigma^*$ die von $\mathcal{M}_f$ berechnete Funktion. Sei $g : \Sigma^* \to \Sigma^*$ die von $\mathcal{M}_g$ berechnete Funktion.
Sei $\mathcal{M}_h$ die vom folgenden Flussdiagramm beschriebene TM und $h : \Sigma^* \to \Sigma^*$ die von $\mathcal{M}_h$ berechnete Funktion.



Ist h die Hintereinanderausführung von f und g, d.h., gilt für alle $w \in \Sigma^*$, dass $g(f(w)) = h(w)$? Zeigen Sie diese Behauptung oder geben Sie ein Gegenbeispiel an.

Aufgabe 4: Eigenschaften von Funktionen 1 Punkt

Aus Ihrem bisherigen Studium kennen Sie bereits die folgenden Definitionen: Eine Funktion $f : X \to Y$ heißt

- *injektiv*, wenn gilt: $\forall x \in X, y \in Y : x \neq y \Rightarrow f(x) \neq f(y)$,
- *surjektiv*, wenn gilt: $\forall y \in Y \ \exists x \in X : f(x) = y$,
- *bijektiv*, wenn sie injektiv und surjektiv ist.

Geben Sie jeweils ein Beispiel für eine Funktion $f : \mathbb{N} \to \mathbb{N}$ an, die

(a) Weder injektiv noch surjektiv ist,

(b) injektiv aber nicht surjektiv ist,

(c) surjektiv aber nicht injektiv ist,

(d) bijektiv ist.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 26. Januar 2018

12. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Gültige Turingmaschinencodierung 1 Punkt

Geben Sie eine kontextfreie Grammatik an, welche die Sprache der codierten Turingmaschinen (s. Kapitel 6.5.1 im Skript) erzeugt.

Aufgabe 2: Turingmaschinendecodierung 2 Punkte

Geben Sie die Turingmaschine an, die zu folgender Codierung (s. Kapitel 6.5.1 im Skript) korrespondiert.

111 0100100101000 11 00101010010 11 00100010001000100 111 00 111

Verwenden Sie dabei die folgende Reihenfolge für die Nummerierung von Zuständen und Bandsymbolen.

Q	q_0, q_1, q_2
Γ	$\sqcup, 0, 1$

Aufgabe 3: Funktionen 4 Punkte

Zeigen Sie, dass die folgenden Aussagen aus der Vorlesung gelten.

(a) Die Funktion stdnum ist bijektiv (Lemma 7.1).

(b) Die Funktion bin2tm ist surjektiv (Lemma 7.2), aber nicht injektiv.

(c) Die Funktion func2power ist surjektiv (Lemma 7.3), aber nicht injektiv.

Aufgabe 4: Charakteristische Funktion 2 Punkte

Definition 7.5 aus der Vorlesung lautet:

> Sei $L \subseteq \Sigma^*$ eine Sprache. Die *charakteristische Funktion* $\chi_L : \Sigma^* \to \{0, 1\}$ ist wie folgt definiert:
>
> $$\chi_L(w) = \begin{cases} 1 & \text{falls } w \in L \\ 0 & \text{falls } w \notin L \end{cases}$$

Satz 7.7 aus der Vorlesung besagt:

> Sei $\{0, 1\} \subseteq \Sigma$. Eine Sprache $L \subseteq \Sigma^*$ ist genau dann entscheidbar, wenn die charakteristische Funktion χ_L berechenbar ist.

Zeigen Sie diese Aussage.

Hinweis: In den folgenden Aufgaben sollen Sie jeweils eine Turingmaschine konstruieren. Sie müssen keine formale Konstruktion der Turingmaschine angeben. Es genügt, wenn Sie Ihre Konstruktionsidee *kurz aber präzise* beschreiben, ggf. unterstützt mit einem Flussdiagramm.
Verwenden Sie jeweils die universelle Turingmaschine $\mathcal{M}_U$.

Aufgabe 5: Abschlusseigenschaften von entscheidbaren Sprachen 1 Punkt
Seien L_1 und L_2 zwei entscheidbare Sprachen über einem Alphabet Σ. Zeigen Sie, dass auch der Schnitt $L_1 \cap L_2$ entscheidbar ist.

Aufgabe 6: Entscheidbarkeit 2 Punkte
Sei $\mathcal{M}$ eine deterministische Turingmaschine. Zeigen Sie, dass es entscheidbar ist, ob die TM $\mathcal{M}$ angesetzt auf das leere Eingabeband jemals ein anderes Zeichen als Blank ($\sqcup$) schreibt.

Aufgabe 7: Simulation 2 Punkte
Betrachten Sie die folgende Sprache.

$$L_{42} = \{\ulcorner \mathcal{M} \urcorner \mid \text{für jede Eingabe } w \text{ der Länge 42 hält } \mathcal{M} \text{ angesetzt auf } w\}$$

Beschreiben Sie eine Turingmaschine $\mathcal{M}_{42}$, die alle Wörter in L_{42} akzeptiert und für alle anderen Wörter nicht hält.

Bemerkung: Sie geben hier ein sogenanntes *Semi-Entscheidungsverfahren* für L_{42} an.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 2. Februar 2018

13. Übungsblatt zur Vorlesung
Informatik III

Empfohlene Vorgehensweise zum Führen eines Reduktionsbeweises:

1. Reduktionsrichtung bestimmen
2. passende Sprache für Reduktion finden
3. Reduktionsfunktion f angeben
4. angeben, dass und ggf. warum f total und berechenbar ist
5. Äquivalenz „$w \in L_1$ gdw. $f(w) \in L_2$“ zeigen
6. verwendete Eigenschaft der gewählten Sprache ((un-/semi-)entscheidbar) angeben und damit die zu zeigende Aussage folgern

Aufgabe 1: Eigenschaften der Reduktionsrelation 2 Punkte

Satz 7.17 der Vorlesung besagt:

> Die Reduktionsrelation $\preceq$ ist reflexiv und transitiv.

Beweisen Sie diese Aussage.

Aufgabe 2: Reduktion I 3 Punkte

Sei $L_1 \subseteq \Sigma^*$ eine entscheidbare Sprache und sei $\emptyset \subsetneq L_2 \subsetneq \Sigma^*$ eine weitere Sprache. Dann gilt $L_1 \preceq L_2$.

(a) Beweisen Sie diese Behauptung.

(b) Für die Fälle $L_2 = \emptyset$ und $L_2 = \Sigma^*$ folgt nicht $L_1 \preceq L_2$.
Beweisen Sie dies für einen der beiden Fälle.

Aufgabe 3: Reduktion II 4 Punkte

Zeigen Sie mit Hilfe von Reduktion, dass die folgende Sprache unentscheidbar ist.

$$H_{\forall w} = \{\ulcorner \mathcal{M} \urcorner \mid \mathcal{M} \text{ angesetzt auf jedes beliebige Wort } w \in \Sigma^* \text{ hält}\}$$

Aufgabe 4: Reduktion III 4 Punkte

Zeigen Sie mit Hilfe von Reduktion, dass die folgende Sprache semi-entscheidbar ist.

$$H_{42} = \{\ulcorner \mathcal{M} \urcorner \mid \mathcal{M} \text{ angesetzt auf die Binärcodierung von 42 hält}\}$$

Verwenden Sie zur Reduktion die semi-entscheidbare Sprache H_ε.

Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 9. Februar 2018

14. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Eigenschaften der Reduktionsrelation 2 Punkte

Beweisen Sie die folgende Aussage.

> Die Reduktionsrelation $\preceq_p$ ist reflexiv und transitiv.

Aufgabe 2: $\mathsf{SAT} \in \mathsf{P}$? 1 Punkt

Während der Erstellung der Aufgaben ist uns folgender Algorithmus eingefallen, der SAT in quadratischer Zeit löst und damit zeigt, dass $\mathsf{P} = \mathsf{NP}$ gilt. Gegeben sei eine boolesche Formel F mit den Variablen $x_1, \ldots, x_k$. Offensichtlich ist $k \leq |F|$.

1. Wir reduzieren das Problem, ob F erfüllbar ist, auf ein einfacheres Problem mit der Formel $F^{(1)}$, die $k-1$ Variablen enthält. Die Reduktion ersetzt jedes Vorkommen von x_k in F einmal durch 0 und einmal durch 1:

   $$F^{(1)} = F[x_k := 0] \vee F[x_k := 1]$$

   Der Algorithmus lässt sich auf einer Zweibandturingmaschine in $3 \cdot |F| + c$ Schritten durchführen, wobei c eine kleine Konstante ist. Insgesamt ist die Reduktion in $O(n)$. Außerdem ist $F^{(1)}$ genau dann erfüllbar, wenn F erfüllbar ist.

2. Wir wiederholen den ersten Schritt k Mal, bis wir eine Formel $F^{(k)}$ erhalten, die keine Variablen mehr enthält. Der Algorithmus hat Laufzeit $k \cdot O(n)$, also $O(n^2)$.

3. Im letzten Schritt berechnen wir den Wahrheitswert von $F^{(k)}$. Weil die Formel keine Variablen mehr enthält, ist das in linearer Zeit möglich. Der gesamte Algorithmus hat also Zeitkomplexität $O(n^2 + n) = O(n^2)$.

Erklären Sie *kurz*, wo der Fehler liegt.

Aufgabe 3: Reduktion 2 Punkte

Welche der folgenden Reduktionen gelten? Beweisen Sie Ihre Behauptungen.

(a) $SAT \preceq H$

(b) $H \preceq SAT$

(c) $SAT \preceq_p H$

(d) $H \preceq_p SAT$

Dabei ist H das allgemeine Halteproblem für Turingmaschinen und SAT das Erfüllbarkeitsproblem für Boolesche Ausdrücke in konjunktiver Normalform.

Aufgabe 4: Polynomielle Reduktion 3 Punkte

Gegeben sei ein Graph $G = (V, E)$. Ein Hamiltonpfad in G ist ein Pfad, der jeden Knoten in V genau einmal besucht.

Das Problem GHP (gerichteter Hamiltonpfad) ist wie folgt definiert:

> *Gegeben:* Ein gerichteter Graph $G = (V, E)$.
>
> *Frage:* Besitzt G einen gerichteten Hamiltonpfad?

Das Problem UHP (ungerichteter Hamiltonpfad) ist wie folgt definiert:

> *Gegeben:* Ein ungerichteter Graph $G = (V, E)$.
>
> *Frage:* Besitzt G einen ungerichteten Hamiltonpfad?

Zeigen Sie: Das Hamiltonpfadproblem für ungerichtete Graphen lässt sich polynomiell auf das Hamiltonpfadproblem für gerichtete Graphen reduzieren, also

$$\text{UHP} \preceq_p \text{GHP}.$$

Es genügt, wenn Sie Ihre Laufzeitabschätzung grob begründen. Sie müssen weder Pseudocode noch Turingmaschinen explizit angeben.

Aufgabe 5: NP-Vollständigkeit 4 Punkte

Das Problem HSET (Hitting Set) ist wie folgt definiert:

> *Gegeben:* Eine Menge M und eine Menge von Teilmengen $\mathcal{S}$ (d.h. $\mathcal{S} = \{S_1, \ldots, S_n\}$, $S_i \subseteq M$ für $i = 1, \ldots, n$) sowie eine natürliche Zahl $k \leq n$.
>
> *Frage:* Gibt es eine Teilmenge $T \subseteq M$, sodass $|T| \leq k$ und $T \cap S_i \neq \emptyset$ für $i = 1, \ldots, n$? Mit anderen Worten: Gibt es eine höchstens k-elementige Teilmenge T, die mit jedem S_i mindestens ein gemeinsames Element hat?

Das NP-vollständige Problem KNÜB (Knotenüberdeckung bzw. Vertex Cover) ist wie folgt definiert.

> *Gegeben:* Ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl $k \leq |V|$.
>
> *Frage:* Besitzt G eine „überdeckende Knotenmenge“ der Größe höchstens k? Eine überdeckende Knotenmenge ist eine Teilmenge $V' \subseteq V$, sodass für alle Kanten $(u, v) \in E$ gilt: $u \in V'$ oder $v \in V'$.

(a) Begründen Sie, warum HSET in NP liegt.

(b) Beweisen Sie, dass HSET NP-vollständig ist. Sie dürfen dabei verwenden, dass KNÜB NP-vollständig ist.

Es genügt, wenn Sie Ihre Laufzeitabschätzung grob begründen. Sie müssen weder Pseudocode noch Turingmaschinen explizit angeben.