**1. Klausur zur Vorlesung**
**Theoretische Grundlagen der Informatik**
**Wintersemester 2019/2020**

# Lösung!

**Beachten Sie:**

- Bringen Sie den Aufkleber mit Ihrem Namen und Ihrer Matrikelnummer auf diesem Deckblatt an und beschriften Sie jedes Aufgabenblatt mit Ihrem Namen und Matrikelnummer.
- Schreiben Sie die Lösungen auf die Aufgabenblätter und Rückseiten. Am Ende der Klausur sind zusätzliche Leerseiten. Fordern Sie zusätzliches Papier bitte nur an, falls Sie den gesamten Platz aufgebraucht haben.
- Es sind keine Hilfsmittel zugelassen.

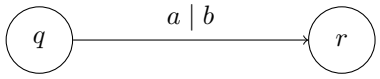
| | Mögliche Punkte | | | | | Erreichte Punkte | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | Σ | a | b | c | d | Σ |
| Aufg. 1 | 2 | 2 | 3 | 3 | 10 | | | | | |
| Aufg. 2 | 3 | 3 | – | – | 6 | | | – | – | |
| Aufg. 3 | 4 | 2 | 3 | – | 9 | | | | – | |
| Aufg. 4 | 3 | 4 | – | – | 7 | | | – | – | |
| Aufg. 5 | 1 | 2 | 3 | 2 | 8 | | | | | |
| Aufg. 6 | 2 | 4 | 4 | – | 10 | | | | – | |
| Aufg. 7 | 2 | 3 | 2 | 3 | 10 | | | | | |
| Σ | | | | | 60 | | | | | |

## **Problem 1:** Endliche Automaten mit Ausgabe 2 + 2 + 3 + 3 = 10 Punkte

In dieser Aufgabe betrachten wir deterministische endliche Automaten, die bei jedem Zustandsübergang ein Zeichen ausgeben. Der Einfachheit halber gehen wir davon aus, dass Eingabe und Ausgabe dasselbe Alphabet $\Sigma$ benutzen. Es gibt zwei Möglichkeiten, die Ausgabe zu definieren:

Bei *Mealy-Automaten* hat jeder Zustandsübergang seine eigene Ausgabe. Ein Mealy-Automat ist also ein 5-Tupel $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ mit Übergangsfunktion $\delta\colon Q \times \Sigma \to Q \times \Sigma$.

**Beispiel:**

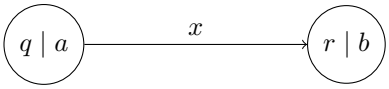



Beim Übergang von $q$ zu $r$ mit gelesenem Zeichen $a$ wird das Zeichen $b$ ausgegeben.

Bei *Moore-Automaten* hängt die Ausgabe nur vom erreichten Zustand ab. Formal ist ein Moore-Automat ein 6-Tupel $\mathcal{A} = (Q, \Sigma, \delta, \lambda, s, F)$ mit Übergangsfunktion $\delta\colon Q \times \Sigma \to Q$ wie üblich und Ausgabefunktion $\lambda\colon Q \to \Sigma$.

*Achtung:* Auch ein Moore-Automat gibt *nur bei einem Zustandsübergang* ein Zeichen aus. Der Startzustand $s$ erzeugt zu Beginn der Abarbeitung noch keine Ausgabe. Wird $s$ aber später durch einen Zustandsübergang wieder erreicht, wird $\lambda(s)$ ausgegeben.

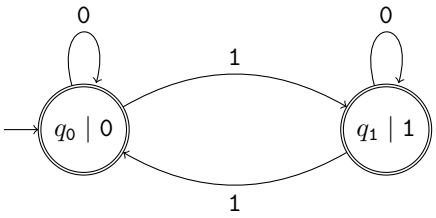
**Beispiel:**




Zustand $q$ gibt beim Erreichen Zeichen $a$ aus, während Zustand $r$ Zeichen $b$ ausgibt.

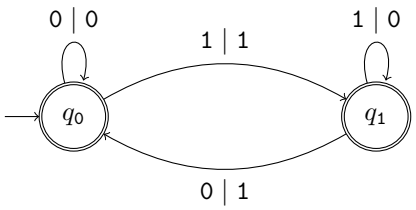
Wir nennen zwei Automaten *äquivalent*, wenn sie für jedes Eingabewort aus $\Sigma^*$ dasselbe Akzeptanzverhalten haben und dieselbe Ausgabe produzieren.

(a) Geben Sie einen äquivalenten Mealy-Automat zu folgendem Moore-Automat an:




(b) Beschreiben Sie, wie für jeden Moore-Automat ein äquivalenter Mealy-Automat konstruiert werden kann. Begründen Sie, warum Ihre Konstruktion korrekt ist!

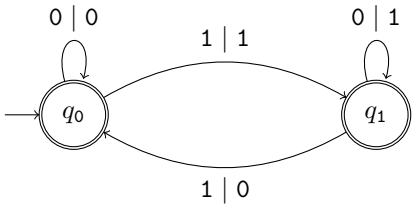
(c) Geben Sie einen äquivalenten Moore-Automat zu folgendem Mealy-Automat an:




(d) Beschreiben Sie, wie für jeden Mealy-Automat ein äquivalenter Moore-Automat konstruiert werden kann. Begründen Sie, warum Ihre Konstruktion korrekt ist!

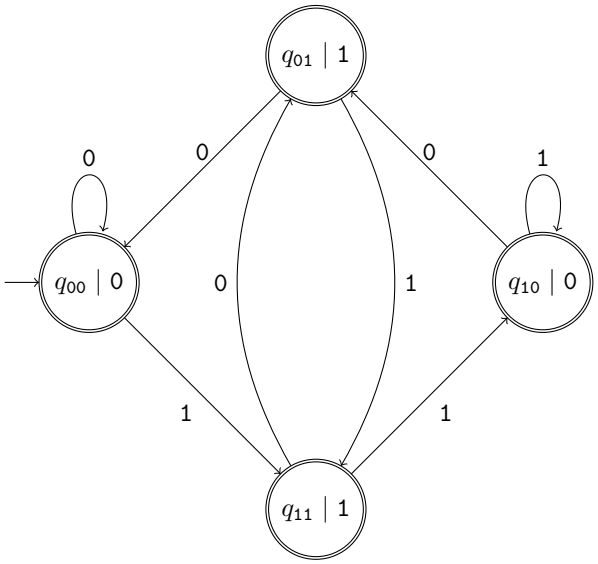
**Lösung:**

(a) Folgender Mealy-Automat ist äquivalent:




(b) Sei ein Moore-Automat $\mathcal{A} = (Q, \Sigma, \delta, \lambda, s, F)$ gegeben. Der Mealy-Automat $\mathcal{A}' = (Q, \Sigma, \delta', s, F)$ mit $\delta'(q, a) = (\delta(q, a), \lambda(\delta(q, a)))$ ist äquivalent. Die Ausgabe wird also von den Zuständen auf ihre eingehenden Übergänge verschoben. Ein Zustand $q$ in $\mathcal{A}$ erzeugt genau dann Ausgabe $\lambda(q)$, wenn er durch einen Zustandsübergang erreicht wird. In $\mathcal{A}'$ erzeugt jeder Übergang zu $q$ genau diese Ausgabe. Also ist das Ausgabeverhalten in beiden Automaten äquivalent.

(c) Folgender Moore-Automat ist äquivalent:




(d) Sei ein Mealy-Automat $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ gegeben. Wähle ein beliebiges Zeichen $x \in \Sigma$. Der Moore-Automat $\mathcal{A}' = (Q \times \Sigma, \Sigma, \delta', \lambda, (s, x), F \times \Sigma)$ mit $\delta'((q, a), b) = \delta(q, b)$ und $\lambda((q, a)) = a$ ist äquivalent. Es wird für jede Kombination aus Zustand $q$ und Zeichen $a$ ein Zustand $(q, a)$ angelegt, der gerade $a$ ausgibt. Ein Übergang, der in $\mathcal{A}$ mit Ausgabe $a$ zu $q$ geführt hat, führt nun zu $(q, a)$. Dadurch wird sichergestellt, dass im erreichten Zustand genau die Ausgabe erzeugt wird, die vorher

die eingehende Kante erzeugt hat. Als Startzustand kann willkürlich $(s, x)$ ausgewählt werden, da der Startzustand zu Beginn keine Ausgabe erzeugt.

## **Problem 2:** Reguläre Sprachen 3 + 3 = 6 Punkte

Für ein endliches Alphabet $\Sigma$ sei eine Sprache der Form

$$L = \{w_k \in \Sigma^* \mid k \in \mathbb{N}^+\}$$

gegeben, wobei für jedes $k \in \mathbb{N}^+$ gilt: $|w_{k+1}| - |w_k| \geq \sqrt[2020]{k}$. In der Wortfolge $w_1, w_2, \ldots$ werden die Längenabstände zwischen zwei aufeinanderfolgenden Wörtern also immer größer.

(a) Zeigen Sie, dass für jedes $n \in \mathbb{N}^+$ ein $k \in \mathbb{N}^+$ existiert, sodass $|w_{k+1}| - |w_k| > n$ und $|w_k| > n$.

(b) Zeigen Sie, dass $L$ nicht regulär ist.

**Lösung:**

(a) Aus $\sqrt[2020]{x} \geq 1$ für $x \geq 1$ folgt $|w_x| \geq x - 1$. Wähle $k = n^{2020} + 2$. Es gilt $|w_k| > n^{2020} \geq n$. Außerdem gilt $|w_{k+1}| - |w_k| \geq \sqrt[2020]{k} = \sqrt[2020]{n^{2020} + 2} > n$.

(b) Wir zeigen, dass $L$ nicht die Aussage des Pumping-Lemmas erfüllt. Wähle für jedes $n \in \mathbb{N}$ das $w_k$ mit $k$ aus Aufgabenteil (a). Betrachte eine Zerlegung $w_k = uvx$ mit $|uv| \leq n$ und $v \neq \varepsilon$. Es ist

$$|v| \leq n \overset{\text{(a)}}{<} |w_{k+1}| - |w_k|.$$

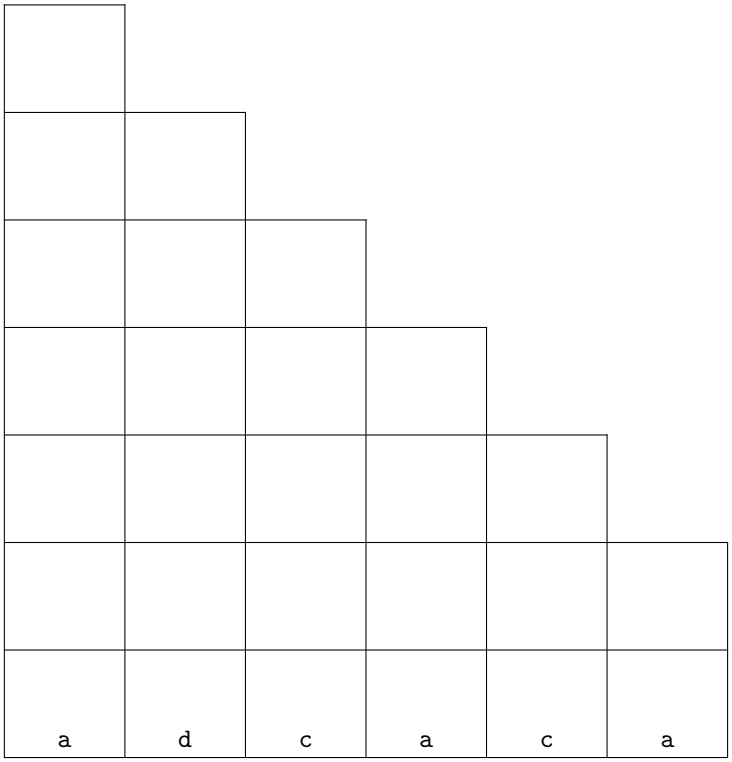
Dann gilt $|uv^2x| = |w_k| + |v| < |w_k| + |w_{k+1}| - |w_k| = |w_{k+1}|$. Also gilt $uv^2x \notin L$.

**Problem 3:** Kontextfreie Grammatiken $4 + 2 + 3 = 9$ Punkte

Gegeben sei die kontextfreie Grammatik $G = (\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}, V = \{A, B, C, D, E\}, A, R)$. Die Regelmenge $R$ enthält folgende Regeln:

$$
\begin{aligned}
A &\to AB \mid EC \mid \mathtt{a} \\
B &\to CA \\
C &\to CC \mid DC \mid \mathtt{c} \\
D &\to \mathtt{d} \\
E &\to CA \mid DA
\end{aligned}
$$

(a) Wenden Sie den CYK-Algorithmus auf das Wort `adcaca` an. Ist das Wort in $L(G)$ enthalten?




(b) Geben Sie einen Ableitungsbaum für das Wort `adcaca` in $G$ an.

(c) Der Ableitungsbaum für ein Wort ist im Allgemeinen nicht eindeutig. Geben Sie ein Verfahren an, das mithilfe der vom CYK-Algorithmus erzeugten Tabelle *alle* möglichen Ableitungsbäume für das überprüfte Wort generiert.

*Hinweis: Berechnen Sie rekursiv für jedes Symbol $X$ in Zelle $V_{ij}$ der Tabelle die Menge aller Ableitungsbäume, die mögliche Ableitungen von $X$ auf das Teilwort $w_{ij}$ repräsentieren.*

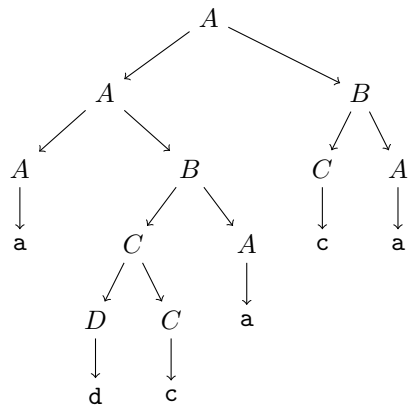
**Lösung:**

(a) Vergleiche folgende Tabelle:

| | | | | | |
|---|---|---|---|---|---|
| $A$ | | | | | |
| | $B, E$ | | | | |
| $A$ | $A, E$ | $B, E$ | | | |
| | $B, E$ | $A$ | $A$ | | |
| | $C$ | $B, E$ | | $B,\ E$ | |
| $A$ | $D$ | $C$ | $A$ | $C$ | $A$ |
| a | d | c | a | c | a |

Das Wort ist also in der Sprache enthalten.

(b) Folgendes ist ein möglicher Ableitungsbaum (es gibt mehrere!):




(c) Für $i = 1$ gibt es genau einen Ableitungsbaum; dieser hat die Form $X \to w_j$. Ansonsten lassen sich die möglichen Ableitungsbäume von $X$ auf $w_{ij}$ rekursiv aus den bereits berechneten Bäumen zusammensetzen. Prüfe dafür für alle $i \leq k < j$, $Y \in V_{ik}$ und $Z \in V_{k+1\,j}$, ob die Regel $X \to YZ$ existiert. Falls ja, müssen die bereits berechneten Bäume für $Y \in V_{ik}$ und $Z \in V_{k+1\,j}$ zusammengesetzt werden. Bei $y$ Bäumen für $Y$ und $z$ Bäumen für $Z$ entstehen $y \cdot z$ mögliche Bäume für $X$. Diese bestehen aus der Wurzel $X$, an die links der Baum für $Y$ und rechts der Baum für $Z$ angehängt wird. Die Ableitungsbäume für $w$ sind dann am Ende die berechneten Bäume für $S \in V_{1n}$.

## **Problem 4:** Turing-Maschinen mit Lesezeichen 3 + 4 = 7 Punkte

Wir erweitern das Berechnungsmodell der Turing-Maschine (ein Kopf, ein Band, eine Spur) folgendermaßen. Zusätzlich zum Kopf verfügt die Turing-Maschine über zwei *Lesezeichen*. Anfangs befinden sich diese auf der Startposition des Kopfes. Neben den üblichen Operationen kann sich die Turing-Maschine nun in jedem Schritt entschließen, eins der Lesezeichen auf die aktuelle Kopfposition zu verschieben. Das Bandsymbol an dieser Position wird dadurch nicht gelöscht.

Außerdem ist es möglich, von der aktuellen Position zu einem beliebigen Lesezeichen zu springen. Sowohl das Verschieben eines Lesezeichens als auch das Springen zu einem Lesezeichen benötigen jeweils einen Zeitschritt. Durch das Springen zu einem Lesezeichen können also größere Distanzen in einem Zeitschritt überwunden werden. Wir bezeichnen dieses Modell als *Lesezeichen-Turing-Maschine* (LTM).

(a) Beschreiben Sie eine LTM, die die Palindromsprache in Linearzeit erkennt.

**Lösung:**
Setze ein Lesezeichen auf dem linkesten Eingabesymbol. Bewege den Kopf nun auf das rechteste Eingabesymbol und setze dort das zweite Lesezeichen. Nun kann in konstanter Zeit zwischen beiden Wortenden hin- und hergesprungen werden. Vergleiche jeweils die beiden Symbole an den Wortenden und verschiebe dann beide Lesezeichen eine Position nach innen.

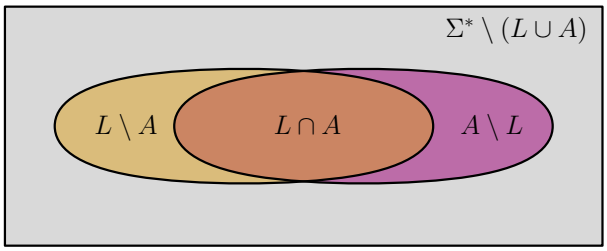
(b) Zeigen Sie, dass Lesezeichen-Turing-Maschinen und (herkömmliche) Turing-Maschinen gleich mächtig sind.

**Lösung:**
Eine Mehrband-TM kann eine LTM leicht simulieren, indem für jedes Lesezeichen der Offset zur aktuellen Kopfposition auf einem zusätzlichen Band gespeichert wird. Dann kann das Springen zu einem Lesezeichen durch Herunter-/Hochzählen dieses Offsets simuliert werden. Aus der Vorlesung ist bekannt, dass Mehrband-TMs und TMs gleich mächtig sind. Umgekehrt kann eine LTM eine TM durch Nichtbenutzen der Lesezeichen simulieren.

## **Problem 5:** Entscheidbarkeit $1 + 2 + 3 + 2 = 8$ Punkte

In dieser Aufgabe sollen Sie eine Eigenschaft von entscheidbaren Sprachen beweisen, die sich informell so ausdrücken lässt: Um zu zeigen, dass eine Sprache entscheidbar ist, reicht es, eine Turing-Maschine zu finden, die für „fast alle“ (d.h. alle bis auf endlich viele) Wörter die richtige Antwort gibt.

Sei dazu eine beliebige Sprache $L$ gegeben und eine Turing-Maschine $\mathcal{M}$, die eine Sprache $A$ entscheidet. Die Situation lässt sich wie folgt darstellen:




(a) Zeigen Sie: Die Menge der entscheidbaren Sprachen ist unter Komplementbildung abgeschlossen.

(b) Zeigen Sie *eine* der folgenden Aussagen:

(1) Die Menge der entscheidbaren Sprachen ist unter Vereinigung abgeschlossen.

(2) Die Menge der entscheidbaren Sprachen ist unter Schnitt abgeschlossen.

Geben Sie explizit an, welche der beiden Aussagen Sie zeigen. Sie dürfen beide Aussagen im Folgenden benutzen.

(c) Zeigen Sie: Wenn $A \subseteq L$ gilt (also $A \setminus L$ leer ist) und $L \setminus A$ endlich ist, dann ist $L$ entscheidbar.

(d) Zeigen Sie: Wenn $L \setminus A$ und $A \setminus L$ endlich sind, dann ist $L$ entscheidbar.

**Lösung:**

(a) Sei $L$ eine entscheidbare Sprache. Es existiert eine deterministische Turing-Maschine, die $L$ entscheidet. Wenn bei dieser das Akzeptanzverhalten vertauscht wird, entscheidet sie $L^c$.

(b) Betrachte zwei entscheidbare Sprachen $L_1$ und $L_2$. Es existieren zwei Turing-Maschinen $\mathcal{M}_1$ und $\mathcal{M}_2$, die $L_1$ bzw. $L_2$ entscheiden. Dann lässt sich $L_1 \cap L_2$ (bzw. $L_1 \cup L_2$) entscheiden, indem erst $\mathcal{M}_1$ und dann $\mathcal{M}_2$ simuliert wird und genau dann akzeptiert wird, wenn beide akzeptieren (bzw. eine der beiden akzeptiert). Da $\mathcal{M}_1$ und $\mathcal{M}_2$ halten, hält auch $\mathcal{M}$.

(c) Nach Aufgabenstellung ist $A$ entscheidbar. $L \setminus A$ ist endlich und somit ebenfalls entscheidbar. Wegen $L = A \cup (L \setminus A)$ und Aufgabenteil (b) ist dann auch $L$ entscheidbar.

(d) $A$, $L \setminus A$ und $A \setminus L$ sind alle entscheidbar. Wir definieren die Abkürzungen $B := A \setminus L$ und $C := L \setminus A$. Es gilt $L = (A \setminus B) \cup C = (A \cap B^c) \cup C$. Da entscheidbare Sprachen unter Schnitt, Komplement und Vereinigung abgeschlossen sind, ist $L$ entscheidbar.

## **Problem 6:** NP-Vollständigkeit $2 + 4 + 4 = 10$ Punkte

In der Vorlesung wurde das NP-vollständige Entscheidungsproblem 3-SAT vorgestellt:

| | |
|---|---|
| **Gegeben:** | Menge $U$ von $n$ Variablen<br>Menge $C$ von $k$ Klauseln über $U$<br>Jede Klausel enthält *höchstens* drei Literale |
| **Frage:** | Existiert eine erfüllende Wahrheitsbelegung für C? |

*Achtung:* In der Vorlesung wurde gefordert, dass jede Klausel *genau* drei Literate enthält. Hier wird nur gefordert, dass jede Klausel *höchstens* drei Literale enthält. Auch in dieser Variante ist 3-SAT NP-vollständig.

Wir betrachten nun das leicht abgewandelte Entscheidungsproblem 3,3-SAT:

| | |
|---|---|
| **Gegeben:** | Menge $U$ von $n$ Variablen<br>Menge $C$ von $k$ Klauseln über $U$<br>Jede Klausel enthält höchstens drei Literale<br>Jede Variable kommt in höchstens drei Klauseln vor |
| **Frage:** | Existiert eine erfüllende Wahrheitsbelegung für C? |

Wir sagen, dass eine Variable $x$ in einer Klausel *vorkommt*, wenn die Klausel $x$ oder $\bar{x}$ enthält. Beachten Sie, dass eine Variable mehrfach in einer Klausel vorkommen kann. Gezählt wird dann nur die Klausel als Ganzes, nicht die Anzahl der Vorkommen innerhalb der Klausel.

(a) Gegeben seien $j$ Variablen $x_1, x_2, \ldots, x_j$. Geben Sie eine Menge $C$ von $j$ Klauseln mit jeweils höchstens drei Literalen an, sodass jede Variable in genau zwei Klauseln vorkommt und $C$ genau zwei erfüllende Belegungen hat: $x_1 = x_2 = ... = x_j =$ `wahr` und $x_1 = x_2 = ... = x_j =$ `falsch`.

(b) Gegeben sei eine 3-SAT-Instanz $(U, C)$, in der genau eine Variable $x$ in mehr als drei Klauseln vorkommt. Konstruieren Sie daraus eine 3,3-SAT-Instanz $(U', C')$ und zeigen Sie, dass diese genau dann erfüllbar ist, wenn $(U, C)$ erfüllbar ist.

(c) Zeigen Sie, dass 3,3-SAT NP-vollständig ist. Geben Sie bei Ihrer Reduktion explizit an, von welchem Problem auf welches reduziert wird!

**Lösung:**

(a) Folgende $j$ Klauseln erfüllen die Forderung:

$$\begin{aligned} x_1 \vee \bar{x}_2 \\ x_2 \vee \bar{x}_3 \\ \cdots \\ x_{j-1} \vee \bar{x}_j \\ x_j \vee \bar{x}_1 \end{aligned}$$

Die Klausel $x_1 \vee \bar{x}_2$ entspricht der Implikation $x_2 \Rightarrow x_1$, d.h. wenn $x_2$ mit `wahr` belegt wird, muss auch $x_1$ mit `wahr` belegt werden. Zusammen bilden die Klauseln die zyklische Implikationskette $x_j \Rightarrow x_{j-1} \Rightarrow \cdots \Rightarrow x_1 \Rightarrow x_j$. Diese erzwingt, dass alle Variablen gleich belegt werden.

(b) Seien $c_1, \ldots, c_j$ die $j > 3$ Klauseln, in denen $x$ vorkommt. Wir ersetzen $x$ durch $j$ neue Variablen $x_1, \ldots, x_j$, wobei wir in Klausel $c_i$ alle Vorkommen von $x$ bzw. $\bar{x}$ durch $x_i$ bzw. $\bar{x}_i$ ersetzen. Zusätzlich fügen wir wie in Aufgabenteil (a) beschrieben $j$ neue Klauseln ein, die erzwingen, dass alle $x_i$ dieselbe Wahrheitsbelegung haben. Die neu eingefügten Variablen kommen jeweils in drei Klauseln vor. Also ist das Ergebnis eine 3,3-SAT-Instanz.

Die konstruierte 3,3-SAT-Instanz $(U', C')$ ist genau dann erfüllbar, wenn die ursprüngliche 3-SAT-Instanz $(U, C)$ erfüllbar ist:

$\Rightarrow$: Betrachte eine Belegung von $U'$, die $C'$ erfüllt. Konstruiere daraus eine Belegung von $U$, die $C$ erfüllt. Für alle Variablen außer $x$ kann der Wahrheitswert einfach übernommen werden. Dann sind offensichtlich alle Klauseln erfüllt, in denen $x$ nicht vorkommt. Die Konstruktion aus Aufgabenteil (a) erzwingt, dass alle $x_i$ denselben Wahrheitswert haben. Übernehme diesen für $x$. Dann sind auch $c_1, \ldots, c_j$ erfüllt.

$\Leftarrow$: Betrachte eine Belegung von $U$, die $C$ erfüllt. Konstruiere daraus eine Belegung von $U'$, die $C'$ erfüllt. Für alle Variablen außer den neu eingefügten $x_1, \ldots, x_j$ kann der Wahrheitswert einfach übernommen werden. Dann sind offensichtlich alle Klauseln erfüllt, in denen $x_1, \ldots, x_j$ nicht vorkommen. Belege jedes neu eingefügte $x_i$ mit dem Wahrheitswert von $x$. Dann sind die modifizierten Klauseln, die vorher $x$ enthielten, offensichtlich erfüllt. Außerdem sind die $j$ neu eingefügten Klauseln erfüllt, da alle $x_i$ denselben Wahrheitswert haben.

(c) Dass 3,3-SAT in NP liegt, folgt trivialerweise daraus, dass 3-SAT in NP liegt. Für eine gegebene Wahrheitsbelegung kann genau auf dieselbe Weise wie bei 3-SAT überprüft werden, ob alle Klauseln erfüllt sind.

Wir zeigen, dass 3,3-SAT NP-schwer ist, indem wir von 3-SAT auf 3,3-SAT reduzieren. Gegeben sei eine 3-SAT-Instanz $(U, C)$. Führe nun für jede Variable $x$, die in mehr als drei Klauseln vorkommt, die Konstruktion aus Aufgabenteil (b) durch. Das Ergebnis ist eine 3,3-SAT-Instanz.

Ein Ersetzungsschritt für eine einzelne Variable $x$ ist in Zeit $\mathcal{O}(k)$ möglich, indem alle $k$ ursprünglichen Klauseln auf Vorkommen von $x$ überprüft werden und dann die $j \leq k$ neuen Variablen und Klauseln generiert werden. In vorigen Ersetzungsschritten eingefügte Klauseln müssen nicht überprüft werden, weil alle darin enthaltenen Variablen in genau zwei Klauseln vorkommen. Insgesamt müssen höchstens $n$ Variablen ersetzt werden, also ist die Reduktion in Zeit $\mathcal{O}(nk)$ durchführbar.

Für jeden einzelnen Ersetzungsschritt wurde bereits in Aufgabenteil (b) gezeigt, dass die Erfüllbarkeit der Instanzen erhalten bleibt. Also gilt dies auch für die gesamte Reduktion.

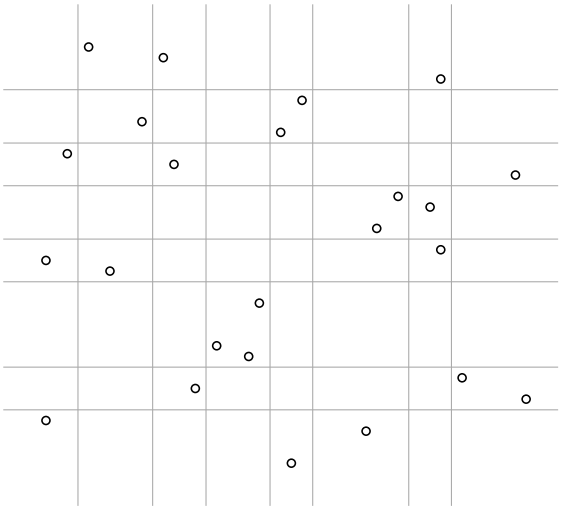
## **Problem 7:** Approximation $2 + 3 + 2 + 3 = 10$ Punkte

Gegeben sei eine Menge $P \subset \mathbb{R}^2$ von $n$ Punkten in der Ebene. Verschiedene Punkte in $P$ haben stets verschiedene $x$-Koordinaten und verschiedene $y$-Koordinaten.

Das Optimierungsproblem MINIMAL-$k$-ENCLOSINGDISK (MkED) besteht darin, das kleinste $r \in \mathbb{R}$ zu bestimmen, sodass gilt: Es gibt einen Kreis mit Radius $r$, der mindestens $k$ Punkte enthält.

Sei $m = \lceil 4n/k \rceil$. Platziere nun $m-1$ vertikale Geraden, die die Ebene so in $m$ vertikale Streifen unterteilen, dass in jedem Streifen höchstens $k/4$ Punkte liegen. Platziere ebenso $m-1$ horizontale Geraden, die die Ebene in $m$ horizontale Streifen unterteilen. Die Geraden sollen außerdem keinen Punkt schneiden.

Hier ist ein Beispiel für $n = 24, k = 12$ und $m = 8$:

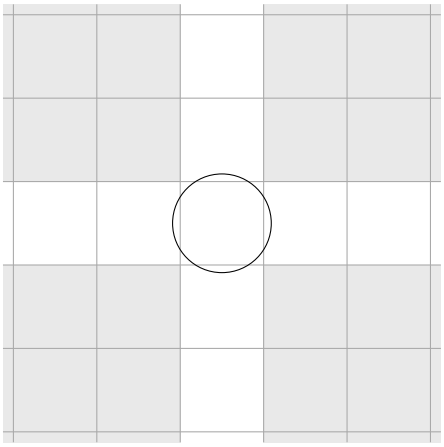


(a) Erklären Sie, wie die Lage der Geraden in Polynomialzeit berechnet werden kann.

**Lösung:**
Sei $x_1, x_2, \ldots, x_n$ die sortierte Folge der $x$-Koordinaten der Punkte in $P$. Für $i = 1, 2, \ldots, m-1$ sei $j = i(k/4)$. Platziere die $i$-te Vertikale zwischen die $x$-Koordinaten $x_j$ und $x_{j+1}$. So schneidet keine Gerade einen Punkt. Das Sortieren erfordert $\mathcal{O}(n \log n)$ Zeit, die restliche Berechnung benötigt Linearzeit. Die Gesamtlaufzeit ist also $\mathcal{O}(n \log n)$.

Die Geraden ergeben ein Gitter, dessen Eckpunkte die Schnittpunkte der Geraden sind.

(b) Zeigen Sie, dass ein Kreis, der keinen Eckpunkt des Gitters enthält (siehe Abbildung), höchstens $k/2$ Punkte aus $P$ enthalten kann.



**Lösung:**
Ein solcher Kreis liegt vollständig innerhalb der Vereinigung eines vertikalen und eines horizontalen Streifens und kann demnach höchstens $k/4 + k/4 = k/2$ Punkte enthalten.

Damit ist gezeigt, dass jeder Kreis, der mindestens $k$ Punkte aus $P$ enthält, auch einen Eckpunkt des Gitters enthält.

Unser Approximationsalgorithmus funktioniert folgendermaßen. Betrachte jeden Gitterpunkt $g$ und berechne den Radius des kleinsten Kreises mit Mittelpunkt $g$, der mindestens $k$ Punkte aus $P$ enthält. Gib den kleinsten so berechneten Radius aus.

(c) Erklären Sie, wie für einen Gitterpunkt $g$ der Radius des kleinsten Kreises mit Mittelpunkt $g$, der mindestens $k$ Punkte aus $P$ enthält, in Polynomialzeit berechnet werden kann.

**Lösung:**
Sortiere die Punkte $P$ aufsteigend nach ihrer Distanz zu $g$. Der gesuchte Radius ist die Entfernung zum $k$-ten Punkt in dieser Folge. Das geht in $\mathcal{O}(n \log n)$ Zeit.

Weil es $(m-1)^2 \in \mathcal{O}((n/k)^2)$ Gitterpunkte gibt, hat also auch der gesamte Approximationsalgorithmus polynomielle Laufzeit.

(d) Zeigen Sie, dass der obige Approximationsalgorithmus eine relative Gütegarantie von 2 hat. Beweisen Sie dazu, dass ein Kreis $C$ mit minimalem Radius $r$ stets vollständig in einem Kreis $C'$ mit Radius $2r$ enthalten ist, wobei der Mittelpunkt von $C'$ ein Eckpunkt des Gitters ist.

*Hinweis: Dreiecksungleichung.*

**Lösung:**
Sei $M$ der Mittelpunkt von $C$. Wir wissen bereits, dass $C$ einen Gitterpunkt $g$ enthält. Wähle $C'$ als den Kreis mit Radius $2r$ und Mittelpunkt $g$. Jeder Punkt $p$ in $C$ hat Abstand höchstens $r$ von $M$. Der Abstand von $M$ zu $g$ ist ebenfalls höchstens $r$. Nach der Dreiecksungleichung ist der Abstand von $p$ zu $g$ höchstens $2r$ und damit ist $p$ in $C$ enthalten.