Karlsruher Institut für Technologie

Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Nachklausur zur Vorlesung
Theoretische Grundlagen der Informatik
Wintersemester 2010/2011

Hier Aufkleber mit Name und Matrikelnr. anbringen

Vorname: ____________________

Nachname: ____________________

Matrikelnummer: ____________________

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 Ihrer Matrikelnummer.
- Schreiben Sie die Lösungen auf die Aufgabenblätter und Rückseiten. Zusätzliches Papier erhalten Sie bei Bedarf von der Aufsicht.
- Zum Bestehen der Klausur sind **20** der möglichen **60** Punkte hinreichend.
- Es sind keine Hilfsmittel zugelassen.

Aufgabe	Mögliche Punkte					Erreichte Punkte				
	a	b	c	d	Σ	a	b	c	d	Σ
1	4	-	-	-	4		-	-	-	
2	3	-	-	-	3		-	-	-	
3	3	3	-	-	6			-	-	
4	4	-	-	-	4		-	-	-	
5	3	4	-	-	7			-	-	
6	5	-	-	-	5		-	-	-	
7	6	-	-	-	6		-	-	-	
8	4	-	-	-	4		-	-	-	
9	1	2	1	1	5					
10	2	4	-	-	6			-	-	
11	10x1				10					
Σ					60					

Aufgabe 1: (4 Punkte)

Beweisen Sie, dass die Sprache $L = \{0^k 10^j \mid k, j \in \mathbb{N}_0, k > j\}$ über dem Alphabet $\Sigma = \{0, 1\}$ nicht regulär ist.

Lösung:
Annahme: L ist regulär. Sei dann n wie im Pumping-Lemma gefordert, d.h., so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung $w = uvx$ mit $|uv| \leq n$, $v \neq \epsilon$, existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$. Betrachte das Wort $w = 0^{n+1}10^n \in L$. Für jede Zerlegung $w = uvx$ mit $|uv| \leq n$, $v \neq \epsilon$ ist $uv^0 x = 0^{n+1-\ell}10^n \notin L$ weil $\ell > 0$. Dies ist ein Widerspruch zum Pumping-Lemma.

Aufgabe 2: (3 Punkte)

Entwerfen Sie eine kontextfreie Grammatik, die die Sprache $\{a^j b^k c^l \mid j, k, l \in \mathbb{N}_0, j > l\}$ über dem Alphabet $\Sigma = \{a, b\}$ erzeugt.

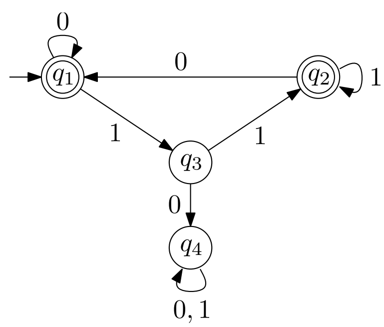
Lösung:
Die Grammatik $G = (\Sigma, V, S, R)$ mit $V = \{S, S', B\}$ und Regeln R wie folgt:

$$\begin{array}{rcl} S & \rightarrow & aS' \\ S' & \rightarrow & aS' | aS'c | B \\ B & \rightarrow & bB | \varepsilon \end{array}$$

Aufgabe 3: (3+3 Punkte)

(a) In einem Wort $w = w_1 \dots w_n$ steht Zeichen x *neben* Zeichen y, wenn es ein $i \in \{1, \dots n-1\}$ gibt, so dass $x = w_i$ und $y = w_{i+1}$ oder $y = w_i$ und $x = w_{i+1}$. Gegeben sei das Alphabet $\Sigma = \{0, 1\}$. Entwerfen Sie einen endlichen Automaten, der genau alle Wörter aus Σ^* akzeptiert, bei denen jede vorkommende 1 neben mindestens einer weiteren 1 steht. Geben Sie dazu das Zustandsübergangsdiagramm des Automaten an.

Lösung:



(b) Ein Wort w' ist *Präfix* eines Wortes w, wenn w sich schreiben lässt als $w = w'u$. Gilt darüber hinaus noch $u \neq \varepsilon$, so ist w' ein *echtes Präfix* von w. Sei L eine reguläre Sprache. Zeigen Sie, dass dann auch $L' = \{w \in L \mid \text{ jedes echte Präfix von } w \text{ ist nicht in } L\}$ regulär ist.

Lösung:
Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ ein deterministischer endlicher Automat, der L akzeptiert. Baue daraus einen Automaten $\mathcal{A}' = (Q \cup \{e\}, \Sigma, \delta', s, F)$, indem ein neuer Fehlerzustand e hinzugefügt wird und δ wie folgt modifiziert wird. Für $a \in \Sigma$ gilt:

$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{für } q \in Q \setminus F \\ e & \text{für } q = e \text{ oder } q \in F \end{cases}$$

Zu zeigen: $L(\mathcal{A}') = L'$

- $\Rightarrow$: Sei $w \in L(\mathcal{A}')$. Da bei Abarbeitung von w in $\mathcal{A}'$ in einem Endzustand endet, können nur Zustandsübergänge benutzt werden, die auch in $\mathcal{A}$ gültig sind, also ist $w \in L$. Nach Konstruktion können dabei keine Übergänge benutzt werden, die aus Endzuständen herausführen, also ist kein echtes Präfix von w in L. Somit ist $w \in L'$.
- $\Leftarrow$: Sei $w \in L'$. Da kein echtes Präfix von w in L ist, werden bei Abarbeitung von w in $\mathcal{A}$ keine Übergänge benutzt, die aus Endzuständen herausführen. Somit akzeptiert auch $\mathcal{A}'$ w.

Aufgabe 4: (4 Punkte)

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit Terminalen $\Sigma = \{a, b, c, d, e\}$, Nichtterminalen $V = \{S, A, B, C, D, E\}$ und Produktionen

$$\begin{aligned} R = \{S &\to DE, \\ A &\to AA|CC|e, \\ C &\to CD|a, \\ D &\to DD|d, \\ E &\to e\} \end{aligned}$$

Überprüfen Sie mit dem Cocke-Younger-Kasami Algorithmus, ob das Wort *aadea* in der Sprache $L(G)$ enthalten ist. Geben Sie dazu alle Zwischenergebnisse an.

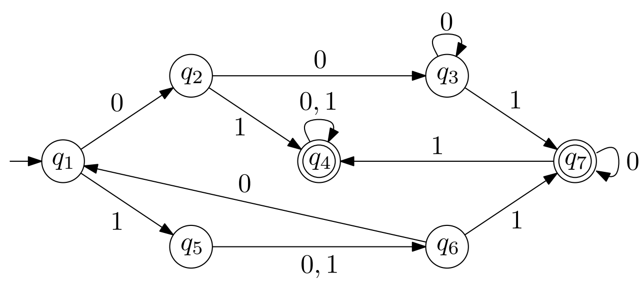
Lösung:

$\emptyset$				
$\{A\}$	$\emptyset$			
$\{A\}$	$\emptyset$	$\emptyset$		
$\{A\}$	$\{C\}$	$\{S\}$	$\emptyset$	
$\{C\}$	$\{C\}$	$\{D\}$	$\{A, E\}$	$\{C\}$
a	a	d	e	a

S ist nicht in V_{15}, deshalb kann *aadea* nicht aus S abgeleitet werden und ist somit nicht in $L(G)$.

Aufgabe 5: (3+4 Punkte)

Gegeben ist folgender endlicher Automat $\mathcal{A}$ über dem Alphabet $\Sigma = \{a, b\}$.



(a) Geben Sie die Sprache $L(\mathcal{A})$ an, die $\mathcal{A}$ akzeptiert. Dazu ist nicht verlangt, dass Sie das Verfahren aus der Vorlesung verwenden.

Lösung:

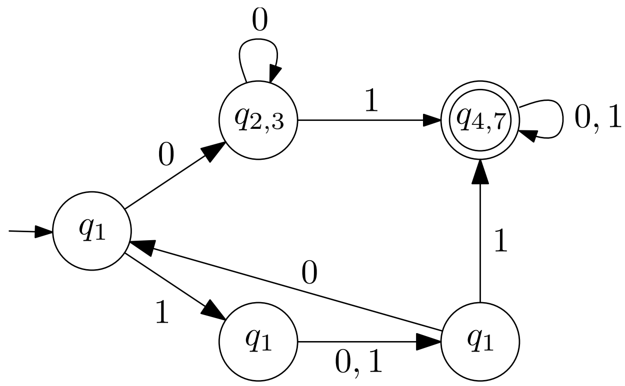
$$L = (\{1\} \cdot \{0,1\} \cdot \{0\})^* \cdot [\{01\} \cdot \{0,1\}^* \ \cup\ \{00\} \cdot \{0\}^* \cdot \{1\} \cdot \{0,1\}^* \ \cup\ \{1\} \cdot \{0,1\} \cdot \{1\} \cdot \{0,1\}^*]$$

(b) Konstruieren Sie den Minimalautomaten zu $\mathcal{A}$. Verwenden Sie dazu ein systematisches Verfahren und geben Sie den zugehörigen Rechenweg an. Sie dürfen dazu benutzen, dass der Minimalautomat 5 Zustände hat.

Lösung:

		$\{q_1, \dots q_7\}$
ϵ	trennt	$\{q_1, q_2, q_3, q_5, q_6\}, \{q_4, q_7\}$
0	trennt	nichts
1	trennt	$\{q_2, q_3, q_6\}, \{q_1, q_5\}$
00	trennt	nichts
01	trennt	$\{q_2, q_3\}, \{q_6\}$
10	trennt	nichts
11	trennt	$\{q_1\}, \{q_5\}$

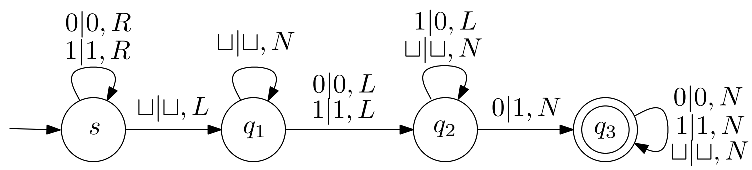
Das sind bereits 5 Äquivalenzklassen, nach Voraussetzung können also keine weiteren Zustände mehr getrennt werden und das Verfahren kann abgebrochen werden. Der Äquivalenzklassenautomat sieht dann wie folgt aus:


0
$q_{2,3}$
1
$q_{4,7}$
0, 1
0
q_1
1
0
1
q_1
0, 1
q_1

Aufgabe 6: (5 Punkte)

Entwerfen Sie eine Turing-Maschine $\mathcal{M}$, die als Eingabe eine positive, binärcodierte Zahl erhält und zu dieser 2 addiert. Sie dürfen annehmen, dass die Eingabezahl mit 01 beginnt. Stellen Sie $\mathcal{M}$ dazu grafisch durch Angabe eines Zustandsdiagrammes dar und bezeichnen Sie den Startzustand mit s. Beschreiben Sie kurz in Worten, wie ihre Turingmaschine arbeitet.

Lösung:



Die Turingmaschine bewegt den Kopf zunächst ans Ende der Eingabe, dann eine Stelle nach links auf das vorletzte Zeichen. Ab jetzt bewegt sich der Kopf immer nur nach links, wenn eine 1 gelesen wird, wird eine Null geschrieben und der Kopf nach links bewegt (Übertrag), die erste gefundene Null wird durch eine 1 überschrieben und die Turingmaschine hält.

Aufgabe 7: (6 Punkte)

Zwei Knoten u und v eines ungerichteten Graphen $G = (V, E)$ heißen *adjazent*, falls sie durch eine Kante verbunden sind, das heißt, falls $\{u, v\}$ in E ist. Eine *Knotenfärbung* von G ist eine Funktion, die jedem Knoten eine Farbe zuordnet.

Problem 4Color

Gegeben:	Ungerichteter Graph $G = (V, E)$
Frage:	Gibt es eine Knotenfärbung von G mit höchstens 4 Farben, so dass je zwei adjazente Knoten verschiedene Farben besitzen?

Zeigen Sie, dass das Problem 4COLOR $\mathcal{NP}$-vollständig ist. Benutzen Sie dazu, dass das Problem 3COLOR NP-vollständig ist:

Problem 3Color

Gegeben:	Ungerichteter Graph $G = (V, E)$
Frage:	Gibt es eine Knotenfärbung von G mit höchstens 3 Farben, so dass je zwei adjazente Knoten verschiedene Farben besitzen?

Lösung:

Problem 4COLOR liegt in $\mathcal{NP}$, denn für eine gegebene Knotenfärbung f eines Graphen G lässt sich in Polynomialzeit überprüfen, ob die Endknoten aller Kanten unterschiedlich gefärbt sind und ob die Zahl der verwendeten Farben höchstens 4 ist.

Um zu zeigen, dass 4COLOR $\mathcal{NP}$-schwer ist, zeigen wir 3COLOR $\propto$ 4COLOR. Gegeben sei eine 3COLOR Instanz $G = (V, E)$. Wir konstruieren die 4COLOR-Instanz G', indem wir zu G einen Knoten hinzunehmen, der mit allen Knoten aus V verbunden ist. Formal ist also $G' = (V \cup \{x\}, E \cup \{\{v, x\} \mid v \in V\})$, wobei x nicht in V enthalten ist.

Dann gilt, dass G genau dann mit 3 Farben färbbar ist, wenn G' mit 4 Farben färbbar ist.

- $\Rightarrow$. Sei f_1 eine zulässige Knotenfärbung von G mit höchstens 3 Farben. Erweitere f_1 zu einer Färbung von G', indem der Knoten x mit einer vierten Farbe gefärbt wird. Dies ist eine gültige Knotenfärbung von G' mit 4 Farben.

- $\Leftarrow$. Sei f_2 eine zulässige Knotenfärbung von G mit höchstens 4 Farben. Die Farbe, die Knoten x zugeordnet ist, kann keinem anderen Knoten zugeordnet sein (da x mit allen übrigen Knoten durch eine Kante verbunden ist). f_2 eingeschränkt auf die Knoten in V ist also eine zulässige Knotenfärbung von G mit höchstens 3 Farben.

- Die Transformation ist offensichtlich polynomial.

Aufgabe 8: (4 Punkte)

Sei L_1 eine nichtentscheidbare Sprache und L_2 eine entscheidbare Sprache mit $L_1 \cap L_2 = \emptyset$.
Zeigen Sie: $L_1 \cup L_2$ ist nicht entscheidbar.

Lösung:
Annahme: $L_1 \cup L_2$ sei entscheidbar. Sei M eine DTM, die $L_1 \cup L_2$ entscheidet und M_2 eine DTM, die L_2 entscheidet. Aus M und M_2 kann wie folgt eine Turingmaschine M_1 konstruiert werden, die L_1 entscheidet:
Sei die Eingabe $w \in \Sigma^*$. M_1 simuliert zunächst M_2 auf w. Falls M_2 akzeptieren würde, ist w in L_2, also nicht in L_1, da $L_1 \cap L_2 = \emptyset$. In diesem Fall verwirft M_1 die Eingabe. Falls M_2 verwerfen würde, ist w nicht in L_2. In diesem Fall simuliert M_1 in der zweiten Phase M auf w. Falls M akzeptieren würde, ist klar, dass $w \in (L_1 \cup L_2) \setminus L_2$ liegt, also in L_1. In diesem Fall akzeptiert M_1 die Eingabe. Falls M nicht akzeptieren würde, ist w nicht in $L_1 \cup L_2$, also auch nicht in L_1, M_1 verwirft die Eingabe.
M_1 hält immer und akzeptiert ein Wort genau dann, wenn es in L_1 liegt. Also entscheidet M_1 L_1.

Aufgabe 9: (1+2+1+1 Punkte)

Sei $\mathcal{A} = (Q, \Sigma, \Gamma, s, Z, \delta, \{f\})$ der Kellerautomat mit Zustandsmenge $Q = \{s, f\}$, Eingabealphabet $\Sigma = \{0, 1\}$, STACK-Alphabet $\Gamma = \{Y, Z\}$, Anfangszustand s, Stack-Initialisierung Z, einzigem Endzustand f und der folgenden Übergangsrelation δ:

$$\begin{aligned} (s, 1, Z) &\mapsto (s, ZY) \\ (s, \varepsilon, Z) &\mapsto (f, \varepsilon) \\ (f, 0, Y) &\mapsto (f, \varepsilon) \end{aligned}$$

(a) Ist $\mathcal{A}$ determininistisch? Begründen Sie Ihre Antwort.

Lösung:
Nein, $\mathcal{A}$ ist nicht deterministisch, da $|\delta(s, 1, Z)| + |\delta(s, \varepsilon, Z)| > 1$.

(b) Dokumentieren Sie eine durch Endzustand akzeptierende Berechung des Wortes 1110. Geben Sie dazu für jeden Schritt die aktuelle Konfiguration an.

Lösung:

- $(s, 1110, Z)$
- $(s, 110, ZY)$
- $(s, 10, ZYY)$
- $(s, 0, ZYYY)$
- $(f, 0, YYY)$
- (f, ε, YY)

(c) Welche Sprache akzeptiert $\mathcal{A}$ durch akzeptierenden Endzustand?

Lösung:
$L = \{1^i 0^j \mid i, j \in \mathbb{N}_0, j \leq i\}$

(d) Welche Sprache akzeptiert $\mathcal{A}$ durch leeren Stack?

Lösung:
$L = \{1^i 0^i \mid i \in \mathbb{N}_0\}$

Aufgabe 10: (2+4 Punkte)

Gegegeben sei eine Grundmenge $S = \{1, \dots, n\}$ und ein Mengensystem $\mathcal{X} \subseteq 2^S$. Jedes Element aus S sei dabei in höchstens 3 Mengen aus $\mathcal{X}$ enthalten.
Ein *Cover* für S ist eine Teilmenge $C \subseteq \mathcal{X}$, so dass jedes $s \in S$ in mindestens einer Menge aus C enthalten ist. Das Problem Cover-3Freq besteht dann darin, ein Cover minimaler Kardinalität zu finden.
Betrachten Sie folgenden Algorithmus:

Algorithmus 1 : Cover-3Freq Approximation

Eingabe : Menge S, $\mathcal{X} \subseteq 2^S$, so dass jedes s aus S in höchstens 3 Mengen aus $\mathcal{X}$ enthalten ist
Ausgabe : Cover C für S

```
C ← ∅ ;
solange S ≠ ∅ tue
    a ← wähle ein Element a ∈ S ;
    C ← C ∪ {X ∈ 𝒳 | a ∈ X} ;
    S ← {s ∈ S | Es gibt kein A ∈ C mit s ∈ A};
```

(a) Gegeben sei die folgende Cover-3Freq Instanz $I = (S, \mathcal{X})$ mit Grundmenge S und Mengensystem $\mathcal{X}$ wie folgt:

$$S = \{1, 2, 3, 4, 5\}, \mathcal{X} = \{\{1, 2, 4\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{3, 5\}\}.$$

Geben Sie ein Cover $C_1 \subseteq \mathcal{X}$ von S minimaler Kardinalität an. Geben Sie zusätzlich das Cover C_2 an, das Algorithmus 1 mit Eingabe I berechnet, falls in jedem Schritt das Element a in S mit der kleinsten Nummer gewählt wird.

Lösung:

- $C_1 = \{\{1, 2, 4\}, \{3, 5\}\}$
- $C_2 = \{\{1, 2, 4\}, \{1, 3\}, \{3, 5\}\}$

(b) Zeigen Sie, dass Algorithmus 1 2-approximativ ist, d.h. dass Algorithmus 1 eine relative Gütegarantie von $2 + 1 = 3$ hat.

Lösung:
Sei $I = (S, \mathcal{X})$ eine Cover-3Freq-Instanz, $\mathcal{A}(I)$ das Cover, das Algorithmus 1 mit Eingabe I berechnet und $\mathrm{OPT}(I)$ eine optimale Lösung von I. Sei k die Anzahl der Schleifendurchläufe, die Algorithmus 1 zur Berechnung von $\mathcal{A}(I)$ benötigt hat und a_i das Element aus S, das beim i-ten Schleifendurchgang ausgewählt wurde. Dann gilt für alle $i, j \in \{1, \dots, k\}, i < j$, dass es kein $X \in \mathcal{X}$ gibt, das sowohl a_i als auch a_j enthält, sonst wäre a_j nicht ausgewählt worden. Da $\mathrm{OPT}(I)$ ein Cover von S, insbesondere also auch von $\{a_1, \dots, a_k\}$ ist, kann jedem a_i ein Element aus $\mathrm{OPT}(I)$ zugeordnet werden, das a_i enthält. Da, wie schon überlegt, jede Menge aus $\mathcal{X}$ höchstens eines der a_i enthalten kann, ist diese Zuordnung eindeutig. Also ist $k \leq |\mathrm{OPT}(I)|$. Weiterhin ist jedes Element aus S höchstens in 3 Mengen aus $\mathcal{X}$ enthalten ist, deshalb werden in jedem Schleifendurchlauf höchstens 3 Mengen zu $\mathcal{A}(I)$ hinzugefügt. Also gilt $|\mathcal{A}(I)| \leq 3 \cdot k$ und folglich

$$\frac{|\mathcal{A}(I)|}{|\mathrm{OPT}(I)|} \leq \frac{3k}{k} = 3$$

Aufgabe 11: (10 Punkte)

Kreuzen Sie für folgende Aussagen an, ob diese wahr oder falsch sind.
Hinweis: Für jede richtige Antwort gibt es einen Punkt, für jede falsche Antwort wird ein Punkt abgezogen. Es wird keine negative Gesamtpunktzahl für diese Aufgabe geben.

Die Sprache $\{a^k a^k \mid k \in \mathbb{N}_0\}$ über dem Alphabet $\Sigma = \{a\}$ ist regulär.

☐ Wahr ☐ Falsch

Lösung: Wahr.

Seien L_1 und L_2 nicht entscheidbar. Dann ist $L_1 \cup L_2$ nicht entscheidbar.

☐ Wahr ☐ Falsch

Lösung: Falsch.

Sei $\mathcal{L}_u := \{wv \in \{0,1\}^* \mid v \in L(T_w)\}$ die universelle Sprache. Dann ist $\mathcal{L}_u^c := \{0,1\}^* \setminus \mathcal{L}_u$ semi-entscheidbar.

☐ Wahr ☐ Falsch

Lösung: Falsch.

Um zu zeigen, dass ein Problem Π in $\mathcal{NP}$ liegt, genügt es, eine polynomielle Transformation von Π auf 3SAT ($\Pi \propto$ 3SAT) anzugeben.

☐ Wahr ☐ Falsch

Lösung: Wahr. Baue NTM für 3SAT um, so dass sie Π löst.

Jede von einem Kellerautomaten durch akzeptierende Endzustände akzeptierte Sprache ist kontextfrei.

☐ Wahr ☐ Falsch

Lösung: Wahr. Siehe Skript.

Sei Π ein Optimierungsproblem. Ein Algorithmus, der Π löst, heißt pseudopolynomiell, falls seine Laufzeit durch ein Polynom der beiden Variablen Eingabegröße und Größe der größten in der Eingabe vorkommenden Zahl beschränkt ist.

☐ Wahr ☐ Falsch

Lösung: Wahr. Siehe Folien und Übungsblatt.

Für jedes Entscheidungsproblem in $\mathcal{P}$ gibt es eine polynomiale Transformation auf 2SAT.

☐ Wahr ☐ Falsch

Lösung: Wahr.

Zu jeder Sprache L_1 gibt es eine Grammatik G mit $L(G) = L_1$.

☐ Wahr ☐ Falsch

Lösung: Falsch. Grammatiken können nur semi-entscheidbare Sprachen erzeugen.

Jede Sprache, die von einem nichtdeterministischen endlichen Automaten akzeptiert wird, ist regulär.

☐ Wahr ☐ Falsch

Lösung: Wahr. NEA's und DEA's sind gleichmächtig.

Ein (polynomiales) Approximationsschema (PAS) für ein Optimierungsproblem Π ist eine Familie von Algorithmen $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$, so dass $\mathcal{A}_\varepsilon$ ein polynomialer Algorithmus ist, der für jede Instanz I von Π einen Wert $\mathcal{A}_\varepsilon(I)$ liefert mit $|\mathrm{OPT}(I) - \mathcal{A}_\varepsilon(I)| \leq \varepsilon$.

☐ Wahr ☐ Falsch

Lösung: Falsch. Siehe Skript.