FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
Professur für Didaktik der Informatik

Lösungs*vorschläge* zu den Staatsexamina:
Theoretische Informatik und Algorithmik

Herbst 2023

Inhaltsverzeichnis

1 Thema

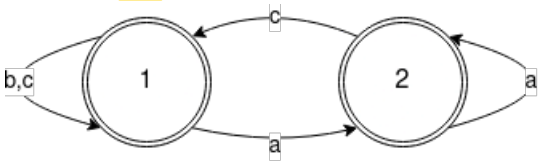
1.1 Teilaufgabe 1: Algorithmik

1.2 Teilaufgabe 2: Theoretische Informatik

1.2.1 Aufgabe 1: Reguläre Sprachen

(a) $\{((a^*|(b|c))^*c)^*(a^*|(b|c)^*)\}$

Erklärung:
Zur Veranschaulichung hier ein DFA, der diese Sprache akzeptiert:



Es müssen beliebige Wörter aus a's, b's oder c's entstehen können, wobei die Möglichkeit verhindert werden muss, dass auf ein a ein b folgt. Die erste wiederholbare Klammer sorgt deshalb für Kreisläufe, die mit einem c enden, so dass wieder ein „Rücksprung" in den Zustand 1 ermöglicht wird. Voraussetzung für ein erneutes Betreten dieses Kreislaufs ist allerdings ein c am Ende des Teilwortes.
Da ein beliebiges Wort aber nicht zwangsläufig auf c enden muss, wird eine weitere Klammer ergänzt, die einen einmaligen „Abschlussdurchlauf" ermöglicht, der nicht auf c endet.

(b) Er akzeptiert die Sprache $L = \{(a^*|a^*ba^*ba^*ba^*b)^*\}$

Begründung:
Anfangs befinden wir uns in einem Endzustand, der beliebig viele weitere a's zulässt. Dafür steht die erste Auswahl innerhalb der Klammer.
Verlassen wir diesen Endzustand über ein b, muss ein Zyklus begonnen werden, der genau vier b's enthält (zwischen denen jeweils beliebig viele a's liegen dürfen), bevor wieder der Entzustand erreicht wird. Für diesen Kreislauf steht die zweite Auswahlmöglichkeit innerhalb der Klammer.

(c) **Übersichtstabelle**

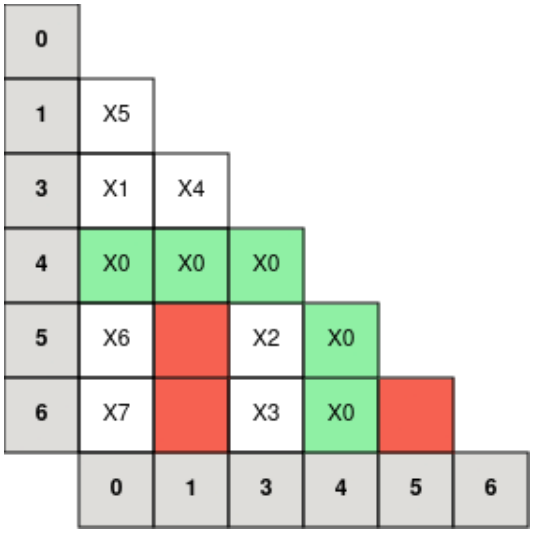
Zustand	**0**	**1**
0	3	1
1	0	6
2	1	4
3	4	4
4	4	5
5	0	5
6	0	5

Zeugentabelle

Zustand	0	1	2	3	4	5	6
Zeuge	ϵ	0	-	1	11	111	01

Der Zustand 2 ist nicht erreichbar, entfällt also im minimierten Automaten.

Tabellenfüllverfahren
4 ist ein Endzustand. Zustandspaare, die einen (nicht *zwei*) von beiden Zuständen enthalten, dürfen wir bereits markieren:



Nebenrechnung

Zustandspaar	0	1	Erläuterung
(3,0)	(4,3)	(4,1)	Eingabe 0: (4,3) führt zu X0. Ergänze X1.
(5,3)	(0,4)	(5,4)	Eingabe 0: (0,4) führt zu X0. Ergänze X2.
(6,3)	(0,4)	(5,4)	Eingabe 0: (0,4) führt zu X0. Ergänze X3.
(3,1)	(4,0)	(4,6)	Eingabe 0: (0,4) führt zu X0. Ergänze X4.
(1,0)	(0,3)	(6,1)	Eingabe 0: (3,0) führt zu X1. Ergänze X5.
(5,0)	(0,3)	(5,1)	Eingabe 0: (3,0) führt zu X1. Ergänze X6.
(6,0)	(0,3)	(5,1)	Eingabe 0: (3,0) führt zu X1.
(6,5)	(0,0)	(5,5)	5 und 6 sind äquivalent.
(6,1)	(0,0)	(5,6)	6 und 1 sind äquivalent.
(5,1)	(0,0)	(5,6)	5 und 1 sind äquivalent.

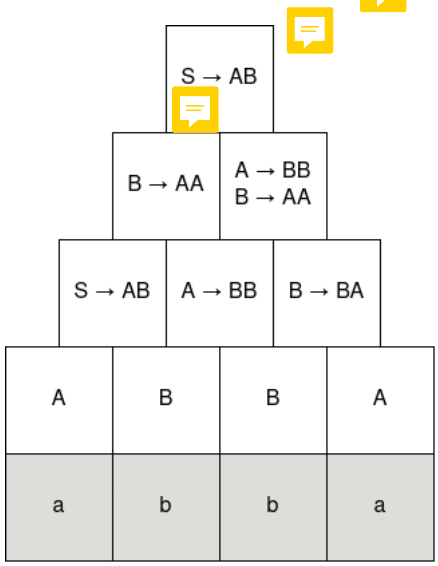
Die Zuständspaare 6 und 1, 5 und 1 sowie 6 und 5 sind unmarkiert geblieben, also bilden 6, 5 und 1 *zusammen* eine Äquivalentsklasse, da alle Eingaben jeweils in äquivalente Zustände führen. Damit bleiben für den minimierten Automaten folgende Äquivalenzklassen:
[1,5,6], 0, 3, 4

Der minimierte Automat A' lautet also:
$A' = (\{[1,5,6], 0, 3, 4\}, \{0,1\}, \delta, s, \{4\})$
mit δ:

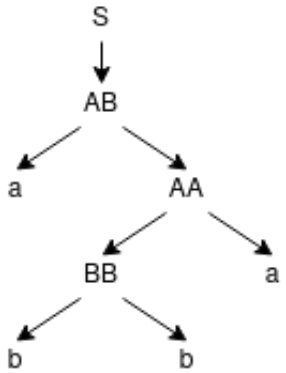
Zustand	0	1
[1,5,6]	0	[1,5,6]
0	3	1
3	4	4
4	4	[1,5,6]

1.2.2 Aufgabe 2: Kontextfreie Sprachen

(a) Ja, das Wort „abba“ ist mit dem CYK-Algorithmus mit [illegible] ableitbar, liegt also in L(G):



Der Ableitungsbaum lässt sich aus der Pyramide bereits ablesen:



(b) Die regulären Sprachen sind eine Teilmenge der kontextfreien Sprachen. Also ist auch R kontextfrei. Kontextfreie Sprachen sind abgeschlossen unter *Verkettung*, denn es lässt sich leicht zeigen:
Sei die Grammatik zu K gegeben durch $G_1 = (N_1, \Sigma_1, P_1, S_1)$ und die Grammatik zu R gegeben durch $G_2 = (N_2, \Sigma_2, P_2, S_2)$. Dann können wir die Verkettung von K und R einfach wie folgt bilden. Für die Verkettungssprache bilden wir eine neue Grammatik, indem wir die Bestandteile der Grammatiken G_1 und G_2 vereinigen, und ergänzen zusätzlich ein neues Startsymbol S, sowie eine neue Produktionsregel $S \to S_1 S_2$. Unsere neue Grammatik, G_3, bildet sich also wie folgt:
$G_3 = (N_1 \cup N_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S_1 \cup [illegible] \cup \{S\})$.

(ii) Dafür muss gelten, dass L kontextfrei ist

Zwar sind auch die kontextsensitiven Sprachen (ein Beispiel dafür ist L') abgeschlossen unter *Verkettung*, jedoch wäre eine Verkettung einer kontextfreien Sprache K mit einer kontextsensitiven Sprache L nicht unbedingt kontextfrei, denn nicht jede kontextsensitive Sprache (etwa L') lässt sich ableiten mit einer kontextfreien Grammatik.

1.2.3 Aufgabe 3: Chomsky-Hierarchie

(a) $L_1 = \{w \in L : vorjedem(stehenmehr[als]\}$

Nichtregularitätsbeweis mit Myhill-Nerode:

> Sei L_1 eine formale Sprache.
> Die binäre Relation $\sim_{L_1} \subseteq \Sigma^*$ ist gegeben durch:
> $x \sim_{L_1} y \Leftrightarrow \forall w \in \Sigma^* : xw \in L \Leftrightarrow yw \in L_1$.
> Ist L_1 regulär, dann ist der Index von $\sim_{L_1}$ endlich.

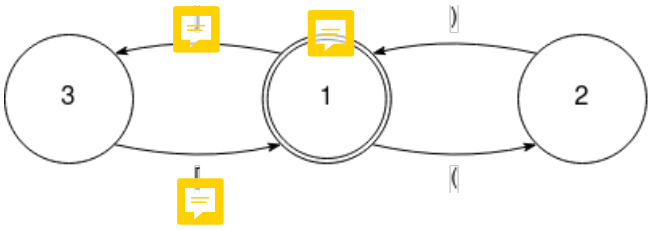
Betrachte die Wörter $[^n($ und $[^k($ mit $k < n$.
Hängt man an sie jeweils das Wort $)^k$ gilt:
$[^n(]^k \in L_1 \forall n, k \in \mathbb{N}$
$[^k(]^k \notin L_1 \forall n, k \in \mathbb{N}$

Durch paarweise gleichmäßiges Erhöhen des Zeichenwiederholungsexponenten erhält man somit unendlich viele Wörter, die sich alle paarweise nicht in derselben Äquivalenzklasse befinden.

$\Rightarrow |\sim_{L_1}| = \infty$
$\Rightarrow L_2$ ist nicht regulär.

(b) $L_2 = \{w \in L :$ auf jede öffnende Klammer folgt direkt eine schließende Klammer $\}$

Zu L_2 lässt sich leicht ein DEA bauen, der sie erkennt:



Damit ist bewiesen, dass L_2 regulär ist, sowie auch, dass L_2 kontextfrei ist, denn die regulären Sprachen sind eine Teilmenge der kontextfreien Sprachen.

1.2.4 Aufgabe 4: Entscheidbarkeit

(a) L_1 ist nicht entscheidbar. Der Satz von Rice kann zum Beweisen der Unentscheidbarkeit von L_1 nicht angewandt werden, denn es handelt sich bei der Spezifikation zu L_1 um keine *semantische* Eigenschaft, sondern um eine *syntaktische*. Stattdessen müssen wir den Beweis führen mit einer Reduktion vom Halteproblem auf das Entscheidbarkeitsproblem von L_1.

	Halteproblem	L_1
Eingabe	DTM M und Wort w	DTM M'
Ja-Instanzen	M hält auf w	DTM M' hält auf ϵ nach $\geq 2^{\lvert\langle N\rangle\rvert}$ Schritten
Nein-Instanzen	M hält nicht auf w	DTM M' hält nicht auf ϵ nach $\geq 2^{\lvert\langle N\rangle\rvert}$ Schritten

Beschreibung der Reduktion

Konstruiere M' wie folgt:

- $Eingabe = \epsilon$? Dann laufe 100 Schritte ohne das Band zu verändern, kehre zurück zum Ausgangspunkt, und verhalte dich wie M auf w.
- $Eingabe \neq \epsilon$? Verhalte dich wie M auf w.

(b) Auch L_2 ist nicht entscheidbar. Wieder können wir den Beweis führen mit einer Reduktion vom Halteproblem auf das Entscheidbarkeitsproblem von L_2, wobei beide Probleme beinahe identisch sind:

	Halteproblem	L_2
Eingabe	DTM M und Wort w	DTM M'
Ja-Instanzen	M hält auf w	DTM M' hält auf Eingabe $\langle N\rangle$
Nein-Instanzen	M hält nicht auf w	DTM M' hält nicht auf Eingabe $\langle N\rangle$

Beschreibung der Reduktion

Konstruiere M' wie folgt:

- M' verhält sich genau wie M, aber für jede verwerfende Stoppkonfiguration von M geht M' stattdessen in eine Endlosschleife.

(c) Auch L_3 ist nicht entscheidbar. Wieder können wir den Beweis führen mit einer Reduktion vom ??? auf das Entscheidbarkeitsproblem von L_3, bei dem es sich um das Problem der Schnittnichtleerheit handelt.

???

2 Thema

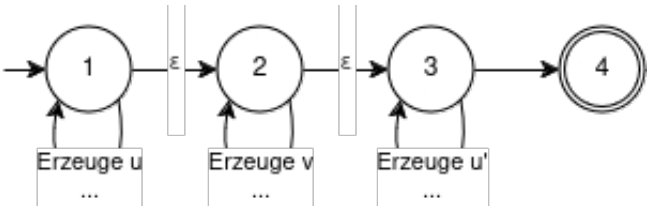
2.1 Teilaufgabe 1: Algorithmik

2.2 Teilaufgabe 2: Theoretische Informatik

2.2.1 Aufgabe 1: Konstruktion

(a) Um aus dem NEA M, der laut Aufgabenstellung ein beliebiger NEA ist, einen NEA M' zu bauen, der L erzeugt, muss M' Wörter der Form $v \in \Sigma*$ akzeptieren, die sowohl ein Präfix $u \in \Sigma*$ als auch ein Suffix $u' \in \Sigma*$ enthalten dürfen.

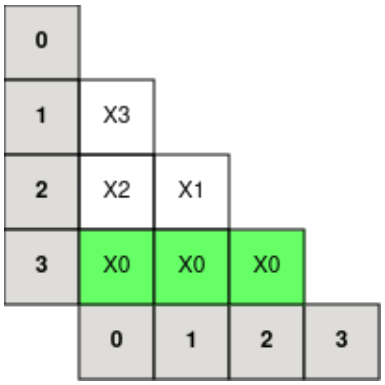
Dafür muss so konstruiert werden, dass es die einzelnen Teilwörter u,v,u' nacheinander erzeugen kann:



(b)

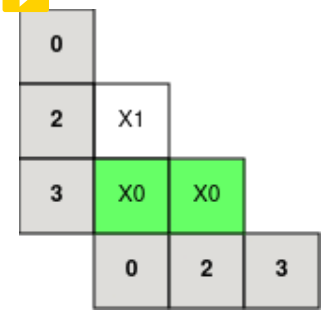
2.2.2 Aufgabe 2: Minimale DEAs

(a) Allgemein eignet sich für die Prüfung der Minimierbarkeit eines DEA das so genannte „Tabellenfüllverfahren“, für das wir eine eine Tabelle aufstellen, anhand der wir Zustandspaare betrachten können. In dieser Tabelle befüllen wir zunächst alle Zellen, deren zugehöriges Zustandspaar einen Endzustand enthält. Von den übrigen Zellen versuchen wir dann, die bereits gefüllten Zellen zu erreichen. Sollte in der treppenförmigen Tabelle eine Zelle ungefüllt bleiben, bedeutet das, dass die beiden Zustände, die zu dieser Zelle gehören, äquivalent sind.
Alle übrigbleibenden Zustände bilden eine Äquivalenzklasse, die zu einem einzigen Zustand zusammengefasst werden können, und damit den vorliegenden DEA minimieren. Wir sehen werden wird für diesen DEA keine Zelle ungefüllt bleiben, was bedeutet, dass in ihm keine äquivalenten Zustände vorliegen, was wiederum bedeutet, dass er bereits minimal ist:



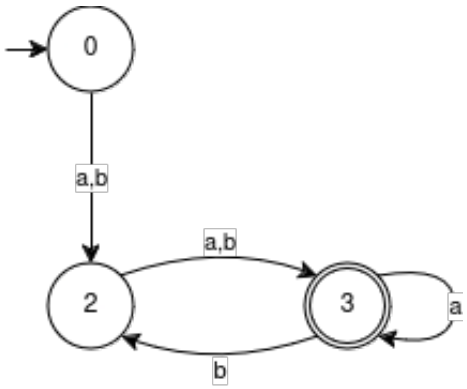
Zustandspaar	a	b	Erläuterung
(2,1)	(0,1)	(3,3)	Eingabe b: (3,3) führt zu X0. Ergänze X1.
(2,0)	(0,1)	(2,1)	Eingabe b: (2,1) führt zu X1. Ergänze X2.
(1,0)	(1,1)	(2,3)	Eingabe b: (2,3) führt zu X0. Ergänze X3.

(b) Hier könnten wir zunächst eine Zeugentabelle aufstellen zwecks Übersichtlichkeit. Jedenfalls können wir erst einmal prüfen, ob überhaupt alle Zustände des DEA erreichbar sind, und dann feststellen, dass Zustand 1 nicht dazu zählt. Damit entfällt Zustand 3 und damit fällt unser Füllverfahren diesmal auch entsprechend kleiner aus:

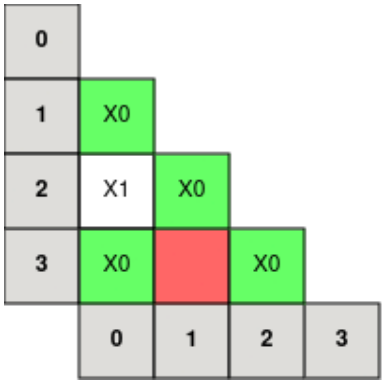


Zustandspaar	a	b	Erläuterung
(1,0)	(0,2)	(3,2)	Eingabe b: (3,2) führt zu X0. Ergänze X1.

Minimierter DEA:

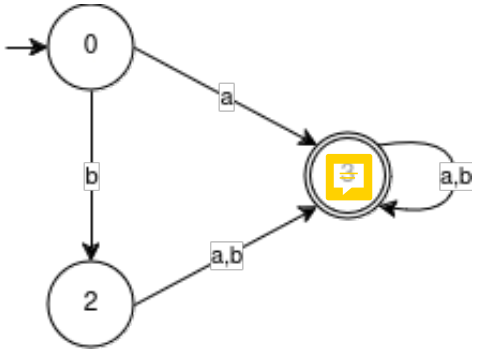


(c) Für diesen Automaten können wir mit dem Tabellenfüllverfahren feststellen, dass eine Zelle ungefüllt bleibt, nämlich die für die Zustände 1 und 3. Diese bilden damit eine Äquivalenzklasse, die im minimierten DEA zu einem einzelnen (End-)Zustand zusammengefasst werden kann:



Zustandspaar	a	b	Erläuterung
(2,0)	(1,3)	(2,3)	Eingabe b: (2,3) führt zu X0. Ergänze X1.
(3,1)	(3,1)	(3,1)	Eingabe b: (2,1) führt zu X1. Ergänze X2.
(3,0)	(1,1)	(2,3)	Eingabe b: (2,3) führt zu X0. Ergänze X3.

Minimierter DEA:



2.2.3 Aufgabe 3: Kontextfreie Grammatiken

(a) Die Umwandlung von G in H erfolgt in vier Schritten:

1) Beseitigung von ϵ-Regeln

Erklärung
Alle Regeln der Form $A \rightarrow \epsilon$ werden beseitigt. Alle Produktionsregeln, die rechts ein Nichtterminal enthalten, das zuvor zu ϵ abgeleitet werden konnte, werden dafür ergänzt um eine Kopie *ohne* dieses Nichtterminalsymbol.
Da das leere Wort aber Teil der Sprache ist, muss es weiter erzeugbar bleiben. Dafür müssen wir einen neuen Startzustand einführen, der zum ursprünglichen Startzustand ableitet sowie zum leeren Wort.

Vorher			Nachher		
S	→	aXY	S0	→	S | ϵ
X	→	Y | ϵ	S	→	aXY | aY
Y	→	aS	X	→	Y
			Y	→	aS | b

2) Beseitigung von Kettenregeln

Erklärung
Jede Produktionsregel der Form A → B mit A,B ∈ V wird bezeichnet als „Kettenregel". Sie stellt einen überflüssigen Zwischenschritt dar, der nicht zur Produktion von Terminalzeichen beiträgt, und darf daher beseitigt werden.

Vorher			Nachher		
S'	→	S | ϵ	S'	→	S | ϵ
S	→	aXY	S	→	aXY
X	→	Y	X	→	aS | b
Y	→	aS | b	Y	→	aS | b

3) Ersetzen von Terminalzeichen

Erklärung
Jedes Terminalzeichen θ, das auftritt in Kombination mit anderen Symbolen, wird ersetzt durch ein neues Nichtterminalzeichen Θ, und die Menge der Produktionsregeln ergänzt um $\theta \rightarrow \Theta$. Wenn dadurch neue Kettenregeln entstehen, müssen diese auch wieder ersetzt werden.

Vorher			Nachher		
S'	→	S | ϵ	S'	→	S | ϵ
S	→	aXY	S	→	AXB
Y	→	aS | b	A	→	a
			B	→	b | AS

4) Beseitigung von Nichtterminalketten

Erklärung
Alle Produktionsregeln, die auf der rechten Seite eine Kette an Nichtterminalzeichen aufweisen, die länger ist als 2, wird zerteilt in Glieder, die zwei Zeichen ersetzen durch ein einzelnes.

Vorher			Nachher		
S'	→	S | ϵ	S'	→	S | ϵ
S	→	AXB	S	→	CB
A	→	a	A	→	a
B	→	b | AS	B	→	b | AS
			C	→	AX

Die Grammatik H wird also spezifiziert wie folgt:

H = (V, Σ, P, S) mit V = {S', S, A, B, C}, Σ = {a, b} und P:

S' → S
S → CB
A → a
B → b | AS
C → AX

(b)

Erklärung
Zwei Einschränkungen gilt es zu beachten: CNF und $|V| \leq 5$.
Die Grammatik darf einzig und allein das Wort „aaaaaaaaaaaa“ erzeugen, also müssen wir die fünf zur Verfügung stehenden Produktionsregeln so gestalten, dass sie sich im einzig möglichen Ableitungsbaum zu einem genau *zwölf*blättrigen Baum verzweigen.
Fünf Variablen bedeuten fünf mögliche Ableitungsschritte, von denen (da CNF) die letzte zwangsläufig auf die Ableitung von Variablen in Terminalsymbole entfällt - es bleiben also vier Ableitungsschritte übrig zum Erreichen unseres „maßgeschneiderten“ Ableitungsbaums.
Eine fortlaufende Verdopplung hätte also $2^4 = 16$ a-Blätter zur Folge - also zu viele. Folglich müssen wir die Verzweigung verknappen. Und dafür bauen wir [illegible] im dritten Ableitungsschritt eine Verzweigungs*bremse* ein, denn die hier entstehende rechte Seite, CD, darf sich nur im C weiter verdoppeln, während die D's schon als Terminalsymbole feststehen. Dadurch produzieren wir nur $2^3 + 2^2 = 8 + 4 = 12$ a-Blätter, und das ist die Anzahl, die wir haben möchten.

[illegible] → AA
A → BB
B → CD
C → DD
D → a

(c) Um die Anforderung der geraden Anzahl an Terminalzeichen zu erreichen, übernehmen wir aus P in P' nur diejenigen Produktionsregeln, die in Ableitungsbäume abgeleitet werden können, deren Blätteranzahl geradzahlig ist.
Die Einschränkung CNF bedeutet dabei, dass wir ein Nichtterminal entweder umwandeln müssen in ein Terminalzeichen, oder in zwei Nichtterminale. Es entfallen in P' also alle Produktionsregeln aus P, die eine beliebige Verzweigung zulassen würden, wie etwa solche der Form $X \rightarrow XY|YY$.

2.2.4 Aufgabe 4: Berechenbarkeit

(a) Ja, L ist semi-entscheidbar. M_1 und M_2 handelt es sich um Turing-Maschinen, daher erzeugen sie Typ-0-Sprachen.
Das Wortproblem für Typ-0-Sprachen ist semi-entscheidbar. Auch L ist eine Typ-0-Sprache, denn es ist sowohl eine Obermenge als auch eine Untermenge zweier anderer Typ-0-Sprachen. Damit ist auch L semi-entscheidbar.

(b) Ja $L = L_2$ ist auch semi-entscheidbar, denn wir können eine Turing-Maschine M' bauen, die $M(L_1)$ und $M(L_2)$ imitiert, und nur anhält, wenn sowohl $M(L_1)$ als auch $M(L_2)$ auf w anhalten.

(c) Ja, L' ist entscheidbar, es gibt also eine Turing-Maschine M', die für jedes Wort w entscheidet, ob es zu L' gehört oder nicht.

Dafür konstruieren wir M' wie folgt:

- Die Turing-Maschine erhält als Eingabe w = uv.
- M' prüft, ob sowohl u als auch v in L gehören, indem sie die Turing-Maschine für L auf beiden Teilen der Eingabe simuliert.
- Falls sowohl u als auch v in L sind, akzeptiert M' w, andernfalls verwirft sie w.

Da beide Teilwörter aus der entscheidbaren Sprache L stammen, wird auch M' für L' richtig entscheiden, ob w = uv in L' liegt oder nicht. Daher ist L' entscheidbar.

(d) Ja, auch L' ist unentscheidbar. L' besteht aus allen Präfixen von Wörtern in L. Wenn L unentscheidbar ist, gibt es keine Möglichkeit, zu entscheiden, ob ein gegebenes Wort u zu L' gehört oder nicht. Das würde nämlich bedeuten, dass wir für jedes Wort u ein entsprechendes Wort v finden müssten, so dass uv in L liegt, was wiederum bedeuten würde, dass L entscheidbar wäre.