Blattsprachen und endliche Automaten

Matthias Galota

11. Mai 1998

Inhaltsverzeichnis

1 Einführung

Bei der Abarbeitung einer Eingabe spannt eine nichtdeterministische polynomialzeitbeschränkte Turingmaschine (NPTM) einen Berechnungsbaum auf. Ordnet man den Blättern des Baumes die auf dem jeweiligen Berechnungsweg erzielte Ausgabe zu, so erhält man eine Folge von Zeichen aus dem Ausgabealphabet – das Blattwort.

Bildet man eine Sprache S über dem Ausgabealphabet, so kann man das Akzeptanzverhalten einer NPTM M mit Hilfe dieser Sprache – die wir *Blattsprache* nennen – definieren, indem man festlegt:

M akzeptiert eine Eingabe, falls das bei der Berechnung entstehende Blattwort in S liegt.

Es ist nun auch möglich, Komplexitätsklassen mit Hilfe von Blattsprachen zu charakterisieren. Dieses Modell wurde unabhängig in [BCS92] und in [Ver93] eingeführt.

Wir wollen nun bestimmte reguläre Blattsprachen und die ihnen zugeordneten Komplexitätklassen betrachten. Dabei nutzen wir aus, daß reguläre Sprachen von endlichen Automaten akzeptiert werden können.

Unser Ziel war, herauszufinden, welche Sprachen über dem Alphabet {0,1} von endlichen Automaten mit zwei bzw. drei Zuständen erkannt werden können und zu erfahren, welchen Komplexitätsklassen sie entsprechen. Doch da es schon 262 relevante endliche Automaten mit drei Zuständen gibt, werden wir nur den Fall der zwei Zustände vollständig abhandeln.

2 Definitionen

Ein Großteil der Definitionen ist [Nie97] entnommen. Wir betrachten im folgenden nur Sprachen über dem Alphabet $\{0, 1\}$.

Der Berechnungsbaum einer NPTM weist nichtdeterministische Verzweigungen auf. Die Anordnung der Pfade an einer Verzweigung hängt nur von der Übergangsfunktion ab. Also kann man die Blätter des Berechnungsbaumes für eine Eingabe mit Hilfe eines nur von der NPTM abhängigen Verfahrens ordnen.

Definition 2.1 Sei M eine NPTM. Sei x eine Eingabe von M. Dann ist

$$\text{leafstring}^M(x)$$

das *Blattwort von M bei Eingabe x* , also die Folge von Ausgaben von M an den Blättern des Berechnungsbaumes, die sich durch dessen Ordnung ergibt.

Wir nennen eine NPTM *balanciert*, wenn der Berechnungsbaum für jede Eingabe x ein vollständiger Binärbaum ist, in dem eventuell ein Teilbaum links außen fehlt.

Üblicherweise wird Balanciertheit so definiert, daß der Teilbaum rechts außen fehlt, aber man kann aus jeder balancierten NPTM, bei deren Berechnungsbäumen ein Teilbaum rechts fehlt, eine konstruieren, bei der der Teilbaum links fehlt, indem man die Anordnung der nichtdeterministischen Verzweigungen invertiert.

Definition 2.2 Sei B eine Sprache. Die Klasse $\mathrm{Leaf}^{\mathrm{P}}(B)$ ist die Klasse aller Sprachen A, für die es eine NPTM M gibt, so daß für alle x gilt:

$$x \in A \quad \leftrightarrow \quad \mathrm{leafstring}^M(x) \in B$$

Die Klasse $\mathrm{BalancedLeaf}^{\mathrm{P}}(B)$ ist die Klasse aller Sprachen A, für die eine NPTM wie oben existiert, die zusätzlich balanciert ist.

Wir können statt einzelner Blattsprachen auch Klassen von Blattsprachen betrachten:

Definition 2.3 Für eine Klasse C ist

$$\begin{aligned} \mathrm{BalancedLeaf}^{\mathrm{P}}(C) &= \textstyle\bigcup_{B \in C} \mathrm{BalancedLeaf}^{\mathrm{P}}(B) \\ \mathrm{Leaf}^{\mathrm{P}}(C) &= \textstyle\bigcup_{B \in C} \mathrm{Leaf}^{\mathrm{P}}(B) \end{aligned}$$

Wir wollen in diesem Zusammenhang nur bestimmte Einschränkungen der Klasse der regulären Blattsprachen betrachten. Um diese Teilklassen zu definieren, benutzen wir einen grundlegenden Satz:

Satz 2.1 Eine Menge ist genau dann regulär, wenn sie von einem endlichen Automaten akzeptiert werden kann.

Wir wollen hier nur deterministische endliche Automaten betrachten. Mit Hilfe der endlichen Automaten können wir nun die folgenden (endlichen) Klassen definieren:

$K_2 :=$ Die Klasse aller regulären Mengen über $\{0, 1\}$, die von endlichen Automaten mit zwei Zuständen akzeptiert werden können.

$K_3 :=$ Die Klasse aller regulären Mengen über $\{0, 1\}$, die von endlichen Automaten mit drei Zuständen akzeptiert werden können.

Unser Ziel ist es nun, zu jeder Blattsprache S aus K_2 oder K_3 die Klassen $\mathrm{Leaf}^{\mathrm{P}}(S)$ und $\mathrm{BalancedLeaf}^{\mathrm{P}}(S)$ zu ermitteln. Als Folgerung daraus wollen wir noch $\mathrm{Leaf}^{\mathrm{P}}(K_2)$ und $\mathrm{BalancedLeaf}^{\mathrm{P}}(K_2)$ bestimmen.

Wir werden nur K_2 vollständig abhandeln, da die Betrachtung von K_3 einen immens hohen Aufwand darstellen würde; es gibt schließlich schon 5832 endliche Automaten der Größe 3. Die Menge der relevanten Automaten können wir zwar auf 262 einschränken, aber das sind noch immer zu viele.

Ein endlicher Automat wird durch ein Tupel (Σ, Z, f, z_0, Z') beschrieben, wobei Σ das Eingabealphabet, Z die Zustandsmenge, f die Übergangsfunktion, z_0 den Startzustand und Z' die Menge der akzeptierenden Zustände darstellt. Alle von uns betrachteten endlichen Automaten mit zwei Zuständen haben folgende Form:

$$E = (\{0,1\}, \{\mathrm{A,B}\}, f_E, \mathrm{A}, Z'_E)$$

Die Automaten mit drei Zuständen besitzen diese Form:

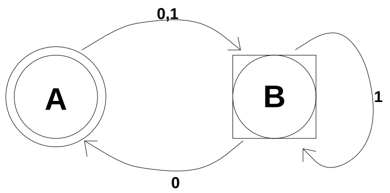
$$E = (\{0,1\}, \{\mathrm{A,B,C}\}, f_E, \mathrm{A}, Z'_E)$$

Also unterscheiden sich verschiedene Automaten gleicher Größe nur in der Übergangsfunktion und in der Menge der akzeptierenden Zustände. Es genügt also zur vollständigen Beschreibung, diese beiden zu kodieren. Das geschieht auf folgende Weise:

Ein Automat der Größe n wird durch ein Wort der Länge $n - 1 + 2 * n$ dargestellt, das aus den ersten n Buchstaben des Alphabets aufgebaut ist. Die ersten $n-1$ Zeichen stellen dabei die Menge der akzeptierenden Zustände dar. Es könnten zwar alle n Zustände akzeptierend sein, doch ein solcher Automat würde einfach jede Eingabe akzeptieren, und es ist müßig, einen Automaten zu betrachten, der solch ein Akzeptanzverhalten besitzt. Gibt es weniger als $n-1$ akzeptierende Zustände, so werden Buchstaben wiederholt. Es bleiben also noch $2*n$ Zeichen übrig, die die Übergangsfunktion kodieren. Da wir nur deterministische endliche Automaten über $\{0,1\}$ betrachten, gibt es in jedem Zustand des Automaten nur zwei mögliche gelesene Zeichen – 0 oder 1. Für jedes dieser Zeichen muß der Automat in einen Zustand springen. Der Übergang beim Lesen einer 0 wird für den k-ten Zustand in Zeichen $n - 1 + (k - 1) * 2$ und der beim Lesen einer 1 in Zeichen $n - 1 + (k - 1) * 2 + 1$ niedergeschrieben. Ein Automat, der z. B. durch das Wort BBBAB beschrieben wird, hat also zwei Zustände.

Sein akzeptierender Zustand ist B : **B**BBAB
Im Zustand A springt er beim Lesen einer 0 in Zustand B : B**B**BAB
Im Zustand A springt er beim Lesen einer 1 in Zustand B : BB**B**AB
Im Zustand B springt er beim Lesen einer 0 in Zustand A : BBB**A**B
Im Zustand B springt er beim Lesen einer 1 in Zustand B : BBBA**B**

Dies ist die graphische Darstellung seiner Übergangsfunktion:



Einige allgemeine Überlegungen sind bei der Betrachtung von Blattsprachen und deren zugehöriger Komplexitätsklassen von Nutzen:

Lemma 2.2 Für eine Sprache S und ihr Komplement $\overline{S}$ gilt:

$$\mathrm{Leaf}^{\mathrm{P}}(\overline{S}) = \mathrm{coLeaf}^{\mathrm{P}}(S)$$

Das entsprechende gilt auch für BalancedLeaf.

Beweis: Der folgende Beweis kann wortwörtlich auch für BalancedLeaf übernommen werden:

Es gibt für jede Menge M aus $\mathrm{Leaf}^{\mathrm{P}}(S)$ eine NPTM T, für die bei Eingabe x und dem dazugehörenden Blattwort w gilt:

$$x \in M \leftrightarrow w \in S$$

Das bedeutet aber:

$$x \in \overline{M} \leftrightarrow w \in \overline{S}$$

Also akzeptiert T mit Hilfe von $\overline{S}$ die Menge $\overline{M}$. Also können wir für jede Menge M aus $\mathrm{Leaf}^{\mathrm{P}}(S)$ eine NPTM finden, die $\overline{M}$ mit Hilfe von $\overline{S}$ akzeptiert. □

Lemma 2.3 Es seien S und T zwei Sprachen, deren symmetrische Differenz nur aus dem leeren Wort besteht. Dann gilt:

$$\mathrm{Leaf}^{\mathrm{P}}(S) = \mathrm{Leaf}^{\mathrm{P}}(T).$$

Das entsprechende gilt auch für BalancedLeaf.

Beweis: Wir betrachten Sprachen nur als Blattsprachen, also an den Blättern eines Berechnungsbaumes. Da ein Berechnungsbaum einer Turingmaschine immer Blätter besitzt, ist es bei der Blattsprache, die eine Klasse definiert, irrelevant, ob sie das leere Wort enthält oder nicht. □

Definition 2.4 Seien S und T Sprachen (wieder über $\{0,1\}$). Wir nennen S die zu T *invertierte Sprache*, falls gilt:

$$w = x_1\, x_2 \ldots x_n \in T \leftrightarrow v = \overline{x_1}\, \overline{x_2} \ldots \overline{x_n} \in S$$

Dabei gilt: $\overline{0} = 1$ und $\overline{1} = 0$. Also sind bei S und T die Rollen von 0 und 1 vertauscht.

Lemma 2.4 Für zwei Sprachen S und T, die zueinander invertiert sind, gilt:

$$\mathrm{Leaf}^{\mathrm{P}}(S) = \mathrm{Leaf}^{\mathrm{P}}(T).$$

Das entsprechende gilt auch für BalancedLeaf.

Beweis: Seien S und T zueinander invertierte Blattsprachen. Zu zeigen sind zwei Inklusionen:

1. $\mathrm{Leaf}^{\mathrm{P}}(S) \subseteq \mathrm{Leaf}^{\mathrm{P}}(T)$:

   Sei $A \in \mathrm{Leaf}^{\mathrm{P}}(S)$ mittels der NPTM M. M' ensteht aus M, indem akzeptierende und ablehnende Zustände vertauscht werden. Nun gilt:

   $x \in A \leftrightarrow$ Das Blattwort von M auf x liegt in S.
   $\leftrightarrow$ Das Blattwort von M' auf x liegt in T.

   Also gilt $A \in \mathrm{Leaf}^{\mathrm{P}}(T)$ mittels M'.

2. $\mathrm{Leaf}^{\mathrm{P}}(T) \subseteq \mathrm{Leaf}^{\mathrm{P}}(S)$:

   Der Beweis ist analog zur Gegenrichtung.

Auch hier ist der Beweis für BalancedLeaf wortwörtlich der gleiche. □

Definition 2.5 Für eine Sprache S definieren wir ihre *Spiegelwortsprache* $R(S)$ mit

$$R(S) := \{w : w^R \in S\}$$

Lemma 2.5 Für eine Sprache S und ihre Spiegelwortsprache $R(S)$ gilt:

$$\mathrm{Leaf}^{\mathrm{P}}(S) = \mathrm{Leaf}^{\mathrm{P}}(R(S)).$$

Das entsprechende gilt auch für BalancedLeaf.

Beweis: Sei S eine Blattsprache. Zu zeigen sind zwei Inklusionen:

1. $\mathrm{Leaf}^{\mathrm{P}}(S) \subseteq \mathrm{Leaf}^{\mathrm{P}}(R(S))$:

   Sei $A \in \mathrm{Leaf}^{\mathrm{P}}(S)$ mittels der NPTM M. M' ensteht aus M, indem die Ordnung der nichtdeterministischen Verzweigungen invertiert wird. Also ist der bei M' auf Eingabe x entstehende Berechnungsbaum spiegelsymmetrisch zu dem von M. Nun gilt:

   $x \in A \leftrightarrow$ Das Blattwort von M auf x liegt in S.
   $\leftrightarrow$ Das Blattwort von M' auf x liegt in $R(S)$.

   Also gilt $A \in \mathrm{Leaf}^{\mathrm{P}}(R(S))$ mittels M'.

2. $\mathrm{Leaf}^{\mathrm{P}}(R(S)) \subseteq \mathrm{Leaf}^{\mathrm{P}}(S)$:

   Der Beweis ist analog zur Gegenrichtung.

Hier muß für BalancedLeaf ein eigener Beweis durchgeführt werden. Zu zeigen sind zwei Inklusionen:

1. $\mathrm{BalancedLeaf}^{\mathrm{P}}(S) \subseteq \mathrm{BalancedLeaf}^{\mathrm{P}}(R(S))$:

   Sei $A \in \mathrm{BalancedLeaf}^{\mathrm{P}}(S)$ mittels der balancierten NPTM M. Die balancierte NPTM M', die $A \in \mathrm{BalancedLeaf}^{\mathrm{P}}(R(S))$ nachweisen soll, simuliert bei Eingabe x zuerst deterministisch den linkesten und den rechtesten Berechnungspfad von M um herauszufinden, welche Höhe der Berechnungsbaum von M auf x besitzt – wir nennen sie n – und ob ein Teilbaum links fehlt. Laufen beide Pfade über gleich viele nichtdeterministische Verzweigungen, so ist der Berechnungsbaum von M vollständig und M' simuliert einfach M spiegelsymmetrisch.

   Ist der Berechnungsbaum von M aber nicht vollständig, fehlt also ein linker Teilbaum, so berechnet M' deterministisch die Größe des fehlenden Teilbaumes – wir nennen sie k. Die Länge des Blattwortes ist also $2^n - k$.

   Nun werden nichtdeterministisch alle Zahlen von 1 bis $2^n - k$ erzeugt. Das Ergebnis ist ein balancierter Berechnungsbaum. Auf Pfad i wird nun der Pfad $2^n - k - 1 + i$ des Berechnungsbaumes von M simuliert.

   Ergab nun die Abarbeitung von x beim Berechnungsbaum von M das Blattwort w, so entsteht bei M' das Blattwort w^R. Somit gilt:

$$\begin{aligned} x \in A &\leftrightarrow w \in S \\ &\leftrightarrow w^R \in R(S) \end{aligned}$$

Also gilt $A \in \mathrm{BalancedLeaf}^{\mathrm{P}}(R(S))$ mittels M'.

2. $\mathrm{BalancedLeaf}^{\mathrm{P}}(R(S)) \subseteq \mathrm{BalancedLeaf}^{\mathrm{P}}(S)$:

   Der Beweis ist analog zur Gegenrichtung. □

Definition 2.6 Seien K_1 und K_2 Klassen von Sprachen. Dann sei

$$\begin{aligned} K_1 \wedge K_2 &:= \{S_1 \cap S_2 : S_1 \in K_1 \wedge S_2 \in K_2\} \\ K_1 \vee K_2 &:= \{S_1 \cup S_2 : S_1 \in K_1 \wedge S_2 \in K_2\} \end{aligned}$$

Diese Schreibweise wurde in [WW85] eingeführt.

Lemma 2.6 Seien S_1 und S_2 Sprachen. Dann gilt:

$$\begin{aligned} \mathrm{Leaf}^{\mathrm{P}}(S_1 \cup S_2) &\subseteq \mathrm{Leaf}^{\mathrm{P}}(S_1) \vee \mathrm{Leaf}^{\mathrm{P}}(S_2) \\ \mathrm{Leaf}^{\mathrm{P}}(S_1 \cap S_2) &\subseteq \mathrm{Leaf}^{\mathrm{P}}(S_1) \wedge \mathrm{Leaf}^{\mathrm{P}}(S_2) \end{aligned}$$

Das entsprechende gilt auch für BalancedLeaf.

Beweis: Zu zeigen sind zwei Inklusionen:

1. $\mathrm{Leaf}^{\mathrm{P}}(S_1 \cup S_2) \subseteq \mathrm{Leaf}^{\mathrm{P}}(S_1) \vee \mathrm{Leaf}^{\mathrm{P}}(S_2)$:

   Sei $A \in \mathrm{Leaf}^{\mathrm{P}}(S_1 \cup S_2)$ mittels der NPTM M. M akzeptiert für jede Blattsprache S eine Sprache aus $\mathrm{Leaf}^{\mathrm{P}}(S)$. Sei B diejenige Sprache aus $\mathrm{Leaf}^{\mathrm{P}}(S_1)$, die von M akzeptiert wird und sei C die Sprache aus $\mathrm{Leaf}^{\mathrm{P}}(S_2)$, die von M akzeptiert wird. Wir müssen nun nur zeigen, daß $A = B \cup C$ gilt. Sei x eine Eingabe von M. Dann gilt:

   $$\begin{aligned} x \in A &\leftrightarrow \text{Das Blattwort von } M \text{ auf } x \text{ liegt in } S_1 \cup S_2. \\ &\leftrightarrow \text{Das Blattwort von } M \text{ auf } x \text{ liegt in } S_1 \text{ oder in } S_2. \\ &\leftrightarrow x \in B \text{ oder } x \in C \\ &\leftrightarrow x \in B \cup C \end{aligned}$$

2. $\mathrm{Leaf}^{\mathrm{P}}(S_1 \cap S_2) \subseteq \mathrm{Leaf}^{\mathrm{P}}(S_1) \wedge \mathrm{Leaf}^{\mathrm{P}}(S_2)$:

   Sei $A \in \mathrm{Leaf}^{\mathrm{P}}(S_1 \cap S_2)$ mittels der NPTM M. Sei B diejenige Sprache aus $\mathrm{Leaf}^{\mathrm{P}}(S_1)$, die von M akzeptiert wird und sei C die Sprache aus $\mathrm{Leaf}^{\mathrm{P}}(S_2)$, die von M akzeptiert wird. Wir müssen nun nur zeigen, daß $A = B \cap C$ gilt. Sei x eine Eingabe von M. Dann gilt:

   $$\begin{aligned} x \in A &\leftrightarrow \text{Das Blattwort von } M \text{ auf } x \text{ liegt in } S_1 \cap S_2. \\ &\leftrightarrow \text{Das Blattwort von } M \text{ auf } x \text{ liegt in } S_1 \text{ und in } S_2. \\ &\leftrightarrow x \in B \text{ und } x \in C \\ &\leftrightarrow x \in B \cap C \end{aligned}$$

Hier ist der Beweis für BalancedLeaf wieder identisch mit dem für Leaf. $\square$

Es gibt Fälle, in denen die Inklusionen echt sind:

Es existiert eine Sprache S mit $\mathrm{Leaf}^{\mathrm{P}}(S) = \mathrm{P}$. Wegen Lemma 2.2 gilt: $\mathrm{Leaf}^{\mathrm{P}}(\overline{S}) = \mathrm{P}$. Es gilt außerdem für jede Komplexitätsklasse K: $K \subseteq K \vee K$. Aus all dem folgt:

$$\begin{array}{rcl} \mathrm{Leaf}^{\mathrm{P}}(S \cup \overline{S}) & = & \mathrm{Leaf}^{\mathrm{P}}(\Sigma^\star) \\ & = & \{\mathbb{N}\} \\ & \subset & \mathrm{P} \\ & \subseteq & \mathrm{P} \vee \mathrm{P} \\ & = & \mathrm{Leaf}^{\mathrm{P}}(S) \vee \mathrm{Leaf}^{\mathrm{P}}(\overline{S}) \end{array}$$

Um gewisse Komplexitätsklassen, die sich ergeben werden, kompakt beschreiben zu können, benötigen wir folgende Definitionen:

Definition 2.7 Seien K_1 und K_2 Komplexitätsklassen. Dann ist $\mathrm{P}^{K_1:K_2}$ wieder eine Komplexitätsklasse und es gilt:

$A \in \mathrm{P}^{K_1:K_2} \leftrightarrow$ Es gibt Mengen $O_1 \in K_1$ und $O_2 \in K_2$ und eine deterministische polynomialzeitbeschränkte TM, die A akzeptiert, wobei sie während der Abarbeitung einer Eingabe zuerst Fragen an das Orakel O_1 und danach an das Orakel O_2 stellen kann.

Für ein $c \in \mathbb{N}$ ist des weiteren $\mathrm{P}^{K_1:K_2[c]}$ eine Komplexitätsklasse und es gilt:

$A \in \mathrm{P}^{K_1:K_2[c]} \leftrightarrow$ Es gibt Mengen $O_1 \in K_1$ und $O_2 \in K_2$ und eine deterministische polynomialzeitbeschränkte TM, die A akzeptiert, wobei sie während der Abarbeitung einer Eingabe zuerst eine nur durch die Anzahl der Rechenschritte beschränkte Anzahl von Fragen an das Orakel O_1 und danach c Fragen an das Orakel O_2 stellen kann.

Definition 2.8 Sei x die Eingabe einer NPTM M. Dann definieren wir $\mathrm{acc}_M(x)$ als die Anzahl aller Pfade im Berechnungsbaum von M auf x, die mit einer Akzeptierung enden.

Definition 2.9 Für eine endliche Menge $S \subset \mathbb{N}$ definieren wir S–P als die Klasse von Mengen A, für die eine NPTM M existiert mit der Eigenschaft

$$x \in A \leftrightarrow \mathrm{acc}_M(x) \in S.$$

Klassen dieser Form wurden erstmals in [GW87] eingeführt. Außerdem setzen wir: 1–NP := $\{1\}$–P. Dieser Spezialfall wurde erstmals in [BG86] eingeführt.

Definition 2.10 Für eine Zahl $k \in \mathbb{N}$ mit $k \geq 2$ und eine endliche Menge $S \subset \mathbb{N}$ mit max$S < k$ definieren wir $S(\text{mod } k)$–P als die Klasse von Mengen A, für die eine NPTM M existiert mit der Eigenschaft

$$x \in A \leftrightarrow \exists n(n \in S \wedge \text{acc}_M(x) \equiv n(\text{mod } k)).$$

Außerdem setzen wir: Mod_kP := co$\{0\}(\text{mod } k)$–P. Diese Klassen wurden erstmals in [CH89] eingeführt.

3 Endliche Automaten mit zwei Zuständen

Wir wollen überprüfen, welche Blattsprachen durch endliche Automaten mit zwei Zuständen akzeptiert werden und welchen Komplexitätsklassen diese entsprechen.

Es existieren $2 * 2^4 = 32$ verschiedene endliche Automaten mit zwei Zuständen, die ein nichttriviales Akzeptanzverhalten besitzen – bei denen also genau ein Zustand akzeptierend ist.

Von diesen müssen wir wiederum 8 Automaten der Form xAAxx nicht betrachten, da diese nie den ersten Zustand verlassen und somit wieder triviale Mengen akzeptieren.

Die restlichen 24 Automaten betrachten wir nun.

3.1 Ergebnisse

n	Automat n	Blattsprache S_n	B'Leaf$^{\mathrm{P}}(S_n)$	Leaf$^{\mathrm{P}}(S_n)$
1	AABAA	$(0 \vee 1)^*0(11)^* \vee (11)^*$	P^{NP}	$\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$
2	BABAA	$(0 \vee 1)^*0(11)^*1 \vee (11)^*1$	P^{NP}	$\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$
3	AABAB	$(0 \vee 1)^*0 \vee \varepsilon$	P	P
4	BABAB	$(0 \vee 1)^*1$	P	P
5	AABBA	$0^*(10^*10^*)^*$	$\oplus$P	$\oplus$P
6	BABBA	$0^*10^*(10^*10^*)^*$	$\oplus$P	$\oplus$P
7	AABBB	0^*	coNP	coNP
8	BABBB	$0^*1(0 \vee 1)^*$	NP	NP
9	ABAAA	$(0 \vee 1)^*1(00)^* \vee (00)^*$	P^{NP}	$\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$
10	BBAAA	$(0 \vee 1)^*1(00)^*0 \vee (00)^*0$	P^{NP}	$\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$
11	ABAAB	$1^*(01^*01^*)^*$	$\oplus$P	$\oplus$P
12	BBAAB	$1^*01^*(01^*01^*)^*$	$\oplus$P	$\oplus$P
13	ABABA	$(0 \vee 1)^*1 \vee \varepsilon$	P	P
14	BBABA	$(0 \vee 1)^*0$	P	P
15	ABABB	1^*	coNP	coNP
16	BBABB	$1^*0(0 \vee 1)^*$	NP	NP
17	ABBAA	$((0 \vee 1)(0 \vee 1))^*$	P	$\oplus$P
18	BBBAA	$((0 \vee 1)(0 \vee 1))^*(0 \vee 1)$	P	$\oplus$P
19	ABBAB	$(0 \vee 1)^*1(00)^*0 \vee (00)^*$	P^{NP}	$\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$
20	BBBAB	$(0 \vee 1)^*1(00)^* \vee (00)^*0$	P^{NP}	$\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$
21	ABBBA	$(0 \vee 1)^*0(11)^*1 \vee (11)^*$	P^{NP}	$\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$
22	BBBBA	$(0 \vee 1)^*0(11)^* \vee (11)^*1$	P^{NP}	$\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$
23	ABBBB	ε	$\{\emptyset\}$	$\{\emptyset\}$
24	BBBBB	$(0 \vee 1)(0 \vee 1)^*$	$\{\mathbb{N}\}$	$\{\mathbb{N}\}$

Wir erhalten also die nichttrivialen Komplexitätsklassen P, NP, coNP, $\oplus$P, P^{NP} und $\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$. Aus dem obigen folgt:

$$\begin{aligned}
\mathrm{BalancedLeaf}^{\mathrm{P}}(K_2) &= \mathrm{P} \cup \mathrm{NP} \cup \mathrm{coNP} \cup \oplus\mathrm{P} \cup \mathrm{P}^{\mathrm{NP}} \\
&= \oplus\mathrm{P} \cup \mathrm{P}^{\mathrm{NP}} \\
\mathrm{Leaf}^{\mathrm{P}}(K_2) &= \mathrm{P} \cup \mathrm{NP} \cup \mathrm{coNP} \cup \oplus\mathrm{P} \cup \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}} \\
&= \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}
\end{aligned}$$

3.2 Beweise

Die Beweise für die Komplextitätsklassen von $S_3 - S_8$, $S_{11} - S_{16}$, S_{23} und S_{24} sind offensichtlich.

Satz 3.1 $\text{BalancedLeaf}^{\text{P}}(S_{17}) = \text{P}$

Beweis: $S_{17} = ((0 \vee 1)(0 \vee 1))^*$
Zu zeigen sind zwei Richungen:

1. $\text{P} \subseteq \text{BalancedLeaf}^{\text{P}}(S_{17})$:

   Sei $A \in \text{P}$. Dann existiert eine deterministische Turingmaschine M, die A entscheidet. Unsere NPTM M' simuliert nun M auf Eingabe x. Falls M x akzeptiert, so verzweigt sich M' in zwei Rechenwege und stoppt in beiden mit einer Akzeptierung. Falls M x aber ablehnt, so stoppt M' mit einer Akzeptierung. So sind alle möglichen Berechnungsbäume von M' vollständige Binärbäume der Höhe 1 oder 2 und der Berechnungsbaum von M' besitzt zwei Blätter, falls $x \in A$, und ein Blatt, falls $x \notin A$.

2. $\text{BalancedLeaf}^{\text{P}}(S_{17}) \subseteq \text{P}$:

   Sei $A \in \text{BalancedLeaf}^{\text{P}}(S_{17})$ mittels NPTM M. Der Berechnungsbaum von M ist also ein vollständiger Binärbaum, bei dem eventuell ein linker Teilbaum fehlt. Ein solcher Binärbaum besitzt nur dann eine ungerade Anzahl von Blättern, wenn zweite Pfad von links eine nichtdeterministische Verzweigung mehr durchlaufen hat als der am weitesten links verlaufende Pfad. Die DTM M' verfolgt also den am weitesten links verlaufenden Berechnungsweg und den jeweils zweiten Pfad von links (nach jeder Verzweigung, die der linkeste Pfad durchläuft, wird ein neuer Pfad zum zweiten von links.) Wird nun die Berechnung auf dem linkesten Pfad beendet, so wird überprüft, ob auf dem dem zweitlinkesten Pfad noch eine nichtdeterministische Verzweigung erfolgt. Ist dies der Fall, so lehnt M' ab.

   Durchlaufen aber beide Pfade gleichviele Verzweigungen, so akzeptiert M'. $\square$

Satz 3.2 $\text{Leaf}^{\text{P}}(S_{17}) = \oplus\text{P}$

Beweis: $S_{17} = ((0 \vee 1)(0 \vee 1))^*$
Zu zeigen sind zwei Richtungen:

1. $\oplus\text{P} \subseteq \text{Leaf}^{\text{P}}(S_{17})$:

   Sei $A \in \oplus\text{P}$ mittels der NPTM M. M' simuliert nun M. Auf jedem Pfad, der mit einer Ablehnung der Eingabe x durch M endet, verzweigt M' in zwei ablehnende Pfade. Wird x aber auf einem Pfad von M

akzeptiert, so hält M' mit einer Akzeptierung. Ob die Anzahl der Pfade von M' gerade ist, hängt also nur davon ab, ob die Anzahl der akzeptierenden Pfade in M gerade ist.

2. $\mathrm{Leaf}^{\mathrm{P}}(S_{17}) \subseteq \oplus\mathrm{P}$:

   Sei $A \in \mathrm{Leaf}^{\mathrm{P}}(S_{17})$ mittels der NPTM M. Die neue NPTM M' simuliert nun M, beendet aber jeden Pfad der Berechnung mit einer Akzeptierung. Also ist die Anzahl der akzeptierenden Pfade im Berechnungsbaum von M' genau die Anzahl der Pfade im Berechnungsbaum von M. $\square$

Für den folgenden Satz benötigen wir ein Hilfslemma:

Lemma 3.3 $A \in \mathrm{P}^{\mathrm{NP}}$ genau dann, wenn ein $B \in \mathrm{P}$ und ein Polynom p existieren, so daß gilt:

$$x \in A \leftrightarrow \max\{y : |y| \leq p(|x|) \wedge (x,y) \in B\} \equiv 1 \pmod 2$$

Beweis: Siehe [Wag90], Satz 6.7 (5). $\square$

Satz 3.4 $\mathrm{BalancedLeaf}^{\mathrm{P}}(S_{10}) = \mathrm{P}^{\mathrm{NP}}$

Beweis: $S_{10} = (0 \vee 1)^*1(00)^*0 \vee (00)^*0$
Zu zeigen sind zwei Richtungen:

1. $\mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{BalancedLeaf}^{\mathrm{P}}(S_{10})$:

   Sei $A \in \mathrm{P}^{\mathrm{NP}}$. Dann existiert also ein $B \in \mathrm{P}$ und ein Polynom p mit $x \in A \leftrightarrow \max\{y : |y| \leq p(|x|) \wedge (x,y) \in B\} \equiv 1 \pmod 2$. Wir gehen hier OBdA davon aus, daß $\max \emptyset = 0$ und $|0| = 0$ gelten. Also kann x nicht in A liegen, wenn kein y mit $(x,y) \in B$ existiert.

   Die nichtdeterministische TM M, die $A \in \mathrm{BalancedLeaf}^{\mathrm{P}}(S_{10})$ nachweisen soll, funktioniert folgendermaßen:

   Bei Eingabe x wird zuerst $p(|x|)$ berechnet. Ist $p(|x|) = 0$, so kann x nicht in A liegen. In diesem Fall stoppt M mit einer Akzeptierung. Dann liegt das Blattwort nicht in S_{10}.

   Ist aber $p(|x|) > 0$, so werden nichtdeterministisch alle Zahlen y mit $|y| \leq p(|x|)$ gebildet. Der Berechnungsbaum, der dabei entsteht, ist ein vollständiger Binärbaum mit $2^{p(|x|)}$ Blättern.

   Da $p(|x|) > 0$ gilt, hat der Berechnungsbaum von M auf x eine gerade Anzahl von Blättern. Es wird auf jedem Pfad geprüft, ob $(x,y) \in B$ gilt. M arbeitet in polynomieller Zeit und es gilt:

$$
\begin{aligned}
x \in A \quad &\leftrightarrow \quad \text{Das maximale } y \text{ mit } (x,y) \in B \text{ ist ungerade.}\\
&\leftrightarrow \quad \text{Mindestens auf einem Pfad wird eine 1 ausgegeben und die am weitesten rechts stehende 1 auf den Blättern des Berechnungsbaumes ist an ungerader Position.}\\
&\leftrightarrow \quad \text{Die Anzahl der Nullen nach der letzten 1 ist ungerade (da die Anzahl der Blätter des Berechnungsbaumes gerade ist.)}\\
&\leftrightarrow \quad \text{Das Blattwort von } M \text{ ist in } S_{10}.
\end{aligned}
$$

Also gilt $A \in \text{BalancedLeaf}^{\text{P}}(S_{10})$ mittels M.

2. $\text{BalancedLeaf}^{\text{P}}(S_{10}) \subseteq \text{P}^{\text{NP}}$:

   Sei $A \in \text{BalancedLeaf}^{\text{P}}(S_{10})$ mittels der TM M. Bei der Abarbeitung einer Eingabe x baut M also einen vollständigen binären Berechnungsbaum auf, bei dem eventuell ganz links ein Teilbaum fehlt. Wir konstruieren nun eine deterministische TM M', die folgendes NP-Orakel O_M benutzt:

   $O_M = \{(x,y) : x \in \mathbb{N}$ und $y \in \{0,1\}^*$ und ein Pfad von M bei der Abarbeitung von x, der durch y beschrieben wird, ist möglich und der Teilbaum über dem von y beschriebenen Pfad enthält eine 1 $\}$

   Zuerst wird geprüft, ob $(x, \varepsilon) \in O_M$ ist, also ob überhaupt eine 1 im Berechnungsbaum von M vorkommt.

   - Ist dies nicht der Fall, so wird geprüft, ob der Berechnungsbaum eine gerade Anzahl von Blättern besitzt. Daß dies deterministisch in polynomieller Zeit möglich ist, zeigt der Beweis von Balanced-$\text{Leaf}^{\text{P}}(S_{17}) \subseteq \text{P}$. Ist die Anzahl gerade, so lehnt ab M', sonst akzeptiert M'.
   - Enthält der Berechnugsbaum mindestens eine 1, so baut M' das y auf, das den Pfad codiert, der zur letzten 1 auf dem Berechnungsbaum führt. Am Anfang gilt: $y = \varepsilon$. Nun überprüft M' in jedem Schritt, ob schon die maximale Verzweigungstiefe auf dem Pfad y erreicht ist.
     - Ist dies der Fall, so hat man die letzte 1 im Berechnungsbaum gefunden. Jetzt wird überprüft, ob $y = \varepsilon$ gilt. Dann besteht das Blattwort nur aus einer einzigen Ziffer – 1 – und liegt somit nicht in S_{19}; also lehnt M' ab. Enthält y aber Zeichen, so wird die letzte Stelle von y abgefragt.

* Ist sie 1, so befindet die letzte 1 in dem Bereich des Berechnungsbaumes, der einem vollständigen Binärbaum entspricht. Da am Schluß des Pfades eine Abzweigung nach rechts genommen wurde, befindet sich rechts von der letzten 1 eine gerade Anzahl von Nullen, also liegt das Blattwort nicht in S_{19}. Somit lehnt M' ab.
* Ist die letzte Stelle von y eine 0, so wurde an der letzten Verzweigung der linke Pfad gewählt. Deshalb befindet sich nach der letzten 1 eine ungerade Anzahl von Nullen und das Blattwort liegt in S_{19} – es sei denn, folgende drei Ereignisse treffen zusammen:

(a) Das y besteht nur aus Nullen, was bedeutet, daß die 1 auf dem linkesten Pfad des Berechnungsbaumes liegt.

(b) Am linken Ende des Berechnungsbaumes fehlt ein Teilbaum.

(c) Der zweite Pfad von links durchläuft eine nichtdeterministische Verzweigung mehr als der linkeste Pfad des Berechnungsbaumes.

In diesem Fall befindet sich rechts von der letzten (und einzigen) 1 eine gerade Anzahl von Nullen und M' muß ablehnen. Dieser Spezialfall ist aber mit deterministisch polynomiellem Aufwand abzufangen.
Ansonsten akzeptiert M'.

– Ist die maximale Verzweigungstiefe noch nicht erreicht, so muß y noch verlängert werden. Dies geschieht, in dem zuerst an y eine 1 angehängt wird. Ist dann immer noch $(x, y) \in O_M$, so ist die letzte 1 im rechten Teilbaum oberhalb der aktuellen Position und man kann mit dem neuen y den nächsten Schritt durchführen. Gilt aber für das neue y: $(x, y) \notin O_M$, so befindet sich die letzte 1 im linken Teilbaum. Dann muß die letzte 1 in y durch eine 0 ersetzt werden. Dann kann man mit dem neuen y weiterrechnen.

Die Anzahl der Fragen an O_M entspricht dann der Höhe des Berechnungsbaumes von M, ist also polynomiell zur Länge der Eingabe. Da für jede Frage nur polynomiell viele Rechenschritte bei M' anfallen, ist also die Berechnung von M' insgesamt polynomialzeitbeschränkt. $\square$

Für den folgenden Satz benötigen wir wieder ein Hilfslemma:

Lemma 3.5 $\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}[1]} = \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$

Beweis: Die Richtung $\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}[1]} \subseteq \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$ ist offensichtlich. Die Gegenrichtung gilt wegen $\oplus\mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$. Dies wurde in [PZ83] bewiesen. $\square$

Satz 3.6 $\mathrm{Leaf}^{\mathrm{P}}(S_{10}) = \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$
In [HH90] wurde die Klasse $\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$ unter dem Namen $\oplus$OptP eingeführt.

Beweis: $S_{10} = (0 \vee 1)^*1(00)^*0 \vee (00)^*0$
Wir zeigen jetzt $\mathrm{Leaf}^{\mathrm{P}}(S_{10}) = \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}[1]}$. Wegen Lemma 3.5 folgt daraus der Satz. Zu zeigen sind zwei Richtungen:

1. $\mathrm{Leaf}^{\mathrm{P}}(S_{10}) \subseteq \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}[1]}$:

   Sei $A \in \mathrm{Leaf}^{\mathrm{P}}(S_{10})$ mittels der TM M. M besitze den maximalen Verzweigungsgrad b. Wir benutzen nun ein NP-Orakel O_M mit

   $O_M = \{(x,y) : x \in \mathbb{N}$ und $y \in \{0,\ldots,b-1\}^*$ und ein Pfad von M bei der Abarbeitung von x, der durch y beschrieben wird, ist möglich und der Teilbaum über dem von y beschriebenen Pfad enthält eine 1 $\}$

   und ein $\oplus$P-Orakel O'_M mit

   $O'_M = \{(x,y) : x \in \mathbb{N}$ und $y \in \{0,\ldots,b-1\}^*$ und ein Pfad von M bei der Abarbeitung von x, der durch y beschrieben wird, ist möglich und der von y beschriebene Pfad hat maximale Verzweigungstiefe und der Teilbaum rechts dieses Pfades enthält eine ungerade Anzahl von Nullen $\}$

   Wir sollten zuerst zeigen, daß O'_M überhaupt eine $\oplus$P-Menge ist:

   Eine NPTM M'' simuliert bei der Eingabe (x,y) die NPTM M, wobei alle Pfade, die links von dem durch y codierten Pfad liegen, nicht beschritten werden. So entspricht der Berechnungsbaum von M'' dem rechts von Pfad y liegenden Teilbaum von M.

   Nun sucht M' die letzte 1 im Blattwort von M auf x mit Hilfe des Orakels O_M. Sie tastet sich dabei Schritt für Schritt durch den Berechnungsbaum von M und fragt bei jeder Verzweigung, in welchen Teilbäumen oberhalb des aktuellen Knotens sich Einsen an den Blättern befinden und verfolgt den rechtesten dieser Pfade. Hat M' so die letzte 1 gefunden, so zählt sie die Anzahl der Blätter rechts von dieser 1 – die ja alle Nullen sind – mit Hilfe einer Frage an O'_M. Ist die Anzahl ungerade, so akzeptiert M' die Eingabe x, ist die Anzahl aber gerade, so wird x abgelehnt.

2. $\mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}[1]} \subseteq \mathrm{Leaf}^{\mathrm{P}}(S_{10})$:

   Sei $A \in \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}[1]}$ mittels der deterministischen TM M, die das NP-Orakel O_M und das $\oplus$P-Orakel O'_M benutzt. Diese Orakel beruhen auf der Auswertung der Berechnungsbäume von NPTMs. Wir nennen diese NPTMs N bzw. N'. Dabei machen wir uns zunutze, daß $\mathrm{Leaf}^{\mathrm{P}}(S_{17}) = \oplus\mathrm{P}$ gilt (mit $S_{17} = ((0 \vee 1)(0 \vee 1))^*$). Bei N' kommt es also nur auf die Anzahl der Blätter des Berechnungsbaumes an, nicht auf die Anzahl der Einsen.

   Die NPTM M', die $A \in \mathrm{Leaf}^{\mathrm{P}}(S_{10})$ beweisen soll, funktioniert folgendermaßen:

   Es wird solange M simuliert, bis die erste Frage „$y_1 \in O_M$?“ auftaucht. Nun wird N mit Eingabe y_1 simuliert. N spannt nun einen Berechnungsbaum auf. An den Blättern des Baumes kann N nun akzeptieren oder ablehnen. Akzeptiert N auf einem Pfad, so verzeigt sich M' in zwei Pfade. Auf dem linken Pfad wird akzeptiert und auf dem rechten Pfad wird N mit der Orakelantwort 1 weitersimuliert. Lehnt N aber ab, so verzweigt M' in zwei Pfade und lehnt auf beiden ab. Eine Ausnahme bildet nur der linkeste Pfad des Berechnungsbaumes von N. Hier wird auf jeden Fall in zwei Pfade verzweigt, auf dem linken Pfad akzeptiert und auf dem rechten Pfad mit der Antwort von N weitergerechnet.

   So wird sichergestellt, daß nur der Pfad, auf dem die richtige Orakelantwort gegeben wurde, für die weitere Berechnung von Bedeutung ist:

   - Ist $y_1 \in O_M$, so gibt es einen rechtesten Pfad von N, der akzeptiert. Auf diesem Pfad wird ja mit der richtigen Orakelantwort weitergerechnet und alle Pfade weiter links werden durch den links vorangestellten Pfad obsolet, da nur die Anzahl der Nullen rechts von der letzten 1 für die Akzeptierung von Bedeutung ist. Alle Pfade mit Nullen (außer dem linkesten) werden durch die doppelte Ablehnung aus der Wertung genommen – es zählt ja am Ende nur, ob die Anzahl der Nullen gerade oder ungerade ist und ein gerader Summand hat auf diese Frage keinen Einfluß.
   - Ist aber $y_1 \notin O_M$, ist der einzige Pfad, auf dem die Berechnung weitergeführt wird, der linkeste. Auf ihm wird nun mit der richtigen Antwort 0 weitergerechnet.

   Bei jeder weiteren Orakelfrage an O_M wird genauso verfahren.

Kommt schließlich die Frage an O'_M, so wird M mit beiden Antworten hintereinander weitersimuliert, und es wird überprüft, ob das Ergebnis von M überhaupt von dieser Frage abhängt.

- Kommt in beiden Fällen dasselbe Ergebnis heraus, so verzweigt sich M' ein letztes Mal in zwei Pfade. Auf dem linken wird akzeptiert und auf dem rechten wird das inverse Ergebnis von M ausgegeben. Es gilt

  $x \in A \leftrightarrow$ Das Ergebnis von M ist 1.
  $\leftrightarrow$ Das entscheidende Teilwort des Blattwortes von M' endet mit einer 0
  $\leftrightarrow$ Das Blattwort von M' liegt in $\mathrm{Leaf}^{\mathrm{P}}(S_{10})$

- Sind die Ergebnisse verschieden, so spaltet sich M' in drei Pfade auf. Auf dem linken wird akzeptiert. Sind die Orakelantworten von O'_M identisch mit den Ergebnissen von M, so wird auf dem mittleren Pfad abgelehnt, und vice versa.

  Auf dem rechten Pfad wird N' simuliert und jeder Pfad wird mit einer Ablehnung abgeschlossen. OBdA seien die Orakelantworten von O'_M identisch mit den Ergebnissen von M. Dann gilt:

  $x \in A \leftrightarrow M$ akzeptiert x.
  $\leftrightarrow$ Die Orakelantwort von O'_M war positiv.
  $\leftrightarrow$ Der Berechnungsbaum vom N' hat eine gerade Anzahl von Blättern. (Da ja $O'_M \in \mathrm{Leaf}^{\mathrm{P}}(S_{17})$ mittels der NPTM N' gilt.)
  $\leftrightarrow$ Der entscheidende Teilbaum von M' endet mit einer ungeraden Zahl von Nullen.
  $\leftrightarrow$ Das Blattwort von M' liegt in $\mathrm{Leaf}^{\mathrm{P}}(S_{10})$

Also gilt $A \in \mathrm{Leaf}^{\mathrm{P}}(S_{10})$ mittels M'. □

Da $S_9 = \overline{S_{10}}$, gilt

- $\mathrm{BalancedLeaf}^{\mathrm{P}}(S_9) = \mathrm{coBalancedLeaf}^{\mathrm{P}}(S_{10}) = \mathrm{coP}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}}$
- $\mathrm{Leaf}^{\mathrm{P}}(S_9) = \mathrm{coP}^{\mathrm{NP}:\oplus\mathrm{P}} = \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$

Da S_1 die Invertierung von S_9 ist, gilt also

- $\mathrm{BalancedLeaf}^{\mathrm{P}}(S_1) = \mathrm{BalancedLeaf}^{\mathrm{P}}(S_9) = \mathrm{P}^{\mathrm{NP}}$
- $\mathrm{Leaf}^{\mathrm{P}}(S_1) = \mathrm{Leaf}^{\mathrm{P}}(S_9) = \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$

Und da S_2 die Invertierung von S_{10} ist, gilt

- $\text{BalancedLeaf}^{\text{P}}(S_2) = \text{BalancedLeaf}^{\text{P}}(S_{10}) = \text{P}^{\text{NP}}$
- $\text{Leaf}^{\text{P}}(S_2) = \text{Leaf}^{\text{P}}(S_{10}) = \text{P}^{\text{NP}:\oplus\text{P}}$

Satz 3.7 $\text{BalancedLeaf}^{\text{P}}(S_{19}) = \text{P}^{\text{NP}}$

Beweis: $S_{19} = (0 \vee 1)^*1(00)^*0 \vee (00)^*$
Da S_{19} große Ähnlichkeit mit S_{10} besitzt, sind auch die beiden Beweise ähnlich. Zu zeigen sind zwei Richtungen:

1. $\text{P}^{\text{NP}} \subseteq \text{BalancedLeaf}^{\text{P}}(S_{19})$:

   Sei $A \in \text{P}^{\text{NP}}$. Dann existiert also ein $B \in \text{P}$ und ein Polynom p mit $x \in A \leftrightarrow \max\{y : |y| \leq p(|x|) \wedge (x,y) \in B\} \equiv 1 (mod 2)$. Wir haben oben schon festgelegt, daß $\max \emptyset = 0$ und $|0| = 0$ gelten. Also kann x nicht in A liegen, wenn kein y mit $(x,y) \in B$ existiert.

   Die nichtdeterministische TM M, die $A \in \text{BalancedLeaf}^{\text{P}}(S_{19})$ nachweisen soll, funktioniert folgendermaßen:

   Bei Eingabe x wird zuerst $p(|x|)$ berechnet. Ist $p(|x|) = 0$, so kann x nicht in A liegen. In diesem Fall stoppt M mit einer Ablehnung. Dann liegt das Blattwort nicht in S_{19}.

   Ist aber $p(|x|) > 0$, so spaltet sich sich der Berechnungsweg von M in zwei Pfade. Auf beiden Pfaden werden nichtdeterministisch alle y mit $|y| \leq p(|x|)$ gebildet. Also existieren nun zwei Teilbäume mit $2^{p(|x|)}$ Blättern die zusammen einen vollständigen Binärbaum mit $2^{p(|x|)+1}$ Blättern ergeben. Im linken Teilbaum stoppt M auf jedem Pfad mit einer Akzeptierung. Im rechten Teilbaum wird jeweils $(x,y) \in B$ getestet.

   Gibt es nun kein Paar (x,y) mit $(x,y) \in B$, so liegt x nicht in A. Dann liegt aber auch das Blattwort nicht in S_{19}, denn es lautet $1^{2^{p(|x|)}}0^{2^{p(|x|)}}$.

   Gibt es aber ein solches Paar, so gilt:

   $x \in A \leftrightarrow$ Das maximale y mit $(x,y) \in B$ ist ungerade.
   $\leftrightarrow$ Die am weitesten rechts stehende 1 auf den Blättern des rechten Teilbaumes ist an ungerader Position.
   $\leftrightarrow$ Die Anzahl der Nullen nach der letzten 1 ist ungerade (da die Anzahl der Blätter des rechten Teilbaumes gerade ist.)
   $\leftrightarrow$ Das Blattwort von M ist in S_{19}.

   Also gilt $A \in \text{BalancedLeaf}^{\text{P}}(S_{19})$ mittels M.

2. $\text{BalancedLeaf}^{\text{P}}(S_{19}) \subseteq \text{P}^{\text{NP}}$:

   Sei $A \in \text{BalancedLeaf}^{\text{P}}(S_{19})$ mittels der TM M. Bei der Abarbeitung einer Eingabe x baut M also einen vollständigen binären Berechnungsbaum auf, bei dem eventuell ganz links ein Teilbaum fehlt. Wir konstruieren nun eine deterministische TM M', die folgendes NP-Orakel O_M benutzt:

   $O_M = \{(x,y) : x \in \mathbb{N}$ und $y \in \{0,1\}^*$ und ein Pfad von M bei der Abarbeitung von x, der durch y beschrieben wird, ist möglich und der Teilbaum über dem von y beschriebenen Pfad enthält eine 1 $\}$

   Zuerst wird geprüft, ob $(x, \varepsilon) \in O_M$ ist, also ob überhaupt eine 1 im Berechnungsbaum von M vorkommt.

   - Ist dies nicht der Fall, so wird geprüft, ob der Berechnungsbaum eine gerade Anzahl von Blättern besitzt. Daß das deterministisch in polynomieller Zeit möglich ist, zeigt der Beweis von Balanced-$\text{Leaf}^{\text{P}}(S_{17}) \subseteq \text{P}$. Ist die Anzahl gerade, so akzeptiert M', sonst lehnt M' ab.
   - Enthält der Berechnugsbaum mindestens eine 1, so entspricht der Beweis exakt dem von $\text{BalancedLeaf}^{\text{P}}(S_{10}) \subseteq \text{P}^{\text{NP}}$. $\square$

Satz 3.8 $\text{Leaf}^{\text{P}}(S_{19}) = \text{P}^{\text{NP}:\oplus\text{P}}$

Beweis: $S_{19} = (0 \vee 1)^* 1 (00)^* 0 \vee (00)^*$
$S_{10} = (0 \vee 1)^* 1 (00)^* 0 \vee (00)^* 0$

Wir zeigen folgende Inklusionen:

1. $\text{Leaf}^{\text{P}}(S_{19}) \subseteq \text{Leaf}^{\text{P}}(S_{10})$:

   Sei $A \in \text{Leaf}^{\text{P}}(S_{19})$ mittels der NPTM M. Wir konstruieren eine NPTM M', die sich bei Eingabe x so verhält: Zuerst verzweigt sie sich in zwei Pfade. Auf dem linken Pfad lehnt sie ab und auf dem rechten Pfad simuliert sie M. Besitzt also der Berechnungsbaum von M mindestens ein Blatt mit einer 1, so spielt der linke Pfad von M' mit der Ablehnung keine Rolle, da nur die letzte 1 im Berechnungsbaum von Bedeutung ist. Also gilt:

   $x \in A \leftrightarrow$ Im Berechnungsbaum von M auf Eingabe x steht hinter der letzten 1 eine ungerade Anzahl von Nullen
   $\leftrightarrow$ Im Berechnungsbaum von M' steht hinter der letzten 1 eine ungerade Anzahl von Nullen
   $\leftrightarrow$ Das Blattwort von M' liegt in S_{10}

Enthält der Berechnungsbaum von M aber nur Blätter mit Nullen, so wird das Blattwort beim Berechnungsbaum von M' um genau eine Null verlängert. Also gilt nun:

$x \in A \leftrightarrow$ Das Blattwort von M auf x besteht aus einer geraden Anzahl von Nullen.
$\leftrightarrow$ Das Blattwort von M' besteht aus einer ungeraden Anzahl von Nullen.
$\leftrightarrow$ Das Blattwort von M' liegt in S_{10}

Also liegt A in $\mathrm{Leaf}^{\mathrm{P}}(S_{10})$ mittels M'.

2. $\mathrm{Leaf}^{\mathrm{P}}(S_{10}) \subseteq \mathrm{Leaf}^{\mathrm{P}}(S_{19})$:

   Die Methode ist exakt identisch zur Gegenrichtung. □

Da $S_{20} = \overline{S_{19}}$, gilt

- $\mathrm{BalancedLeaf}^{\mathrm{P}}(S_{20}) = \mathrm{coBalancedLeaf}^{\mathrm{P}}(S_{19}) = \mathrm{coP}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}}$
- $\mathrm{Leaf}^{\mathrm{P}}(S_{20}) = \mathrm{coP}^{\mathrm{NP}:\oplus\mathrm{P}} = \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$

Da S_{21} die Invertierung von S_{19} ist, gilt also

- $\mathrm{BalancedLeaf}^{\mathrm{P}}(S_{21}) = \mathrm{BalancedLeaf}^{\mathrm{P}}(S_{19}) = \mathrm{P}^{\mathrm{NP}}$
- $\mathrm{Leaf}^{\mathrm{P}}(S_{21}) = \mathrm{Leaf}^{\mathrm{P}}(S_{19}) = \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$

Und da S_{22} die Invertierung von S_{20} ist, gilt

- $\mathrm{BalancedLeaf}^{\mathrm{P}}(S_{22}) = \mathrm{BalancedLeaf}^{\mathrm{P}}(S_{20}) = \mathrm{P}^{\mathrm{NP}}$
- $\mathrm{Leaf}^{\mathrm{P}}(S_{22}) = \mathrm{Leaf}^{\mathrm{P}}(S_{20}) = \mathrm{P}^{\mathrm{NP}:\oplus\mathrm{P}}$

4 Endliche Automaten mit drei Zuständen

4.1 Bestimmung der Anzahl relevanter Automaten

Bei deterministischen endlichen Automaten über {0,1} mit drei Zuständen gibt es $3^{2*3} = 729$ verschiedene Übergangsfunktionen. Außerdem existieren für jede Übergangsfunktion noch folgende Mengen von akzeptierenden Zuständen: $\emptyset$, {A}, {B}, {C}, {A,B}, {A,C}, {B,C} und {A,B,C}. Demnach ist die Anzahl aller endlichen Automaten mit drei Zuständen $729 * 8 = 5832$.

Man muß aber nicht alle diese Automaten betrachten. Die Anzahl der relevanten Automaten kann in mehreren Schritten auf 262 gesenkt werden.

4.1.1 Automaten, die den ersten Zustand nicht verlassen

Ein endlicher Automat der Form xxAAxxxx verläßt den ersten Zustand nie. Also akzeptiert er entweder die Sprache $(0 \vee 1)^*$ (falls sein erster Zustand akzeptierend ist), oder er akzeptiert $\emptyset$.

Es gibt $3^4 * 8 = 648$ Automaten dieses Typs, die nicht betrachtet werden müssen. Es verbleiben noch $5832 - 648 = 5184$.

4.1.2 Automaten die nur zwei Zustände benutzen

Ein Automat mit drei Zuständen, bei dem ein Zustand vom Startzustand aus nicht erreicht werden kann, verhält sich nicht anders als ein Automat mit zwei Zuständen. Diesen Fall haben wir aber schon behandelt. Automaten mit folgenden Kodierungen benutzen nur zwei Zustände:

xxABAAxx	xxABABxx	xxABBAxx	xxABBBxx
xxBAAAxx	xxBAABxx	xxBABAxx	xxBABBxx
xxBBAAxx	xxBBABxx	xxBBBAxx	xxBBBBxx
xxACxxAA	xxACxxAC	xxACxxCA	xxACxxCC
xxCAxxAA	xxCAxxAC	xxCAxxCA	xxCAxxCC
xxCCxxAA	xxCCxxAC	xxCCxxCA	xxCCxxCC

Davon gibt es $4 * 6 * 3^2 * 8 = 1728$ Exemplare. Da alle diese Automaten den ersten Zustand verlassen, gibt es keine Überschneidungen mit 4.1.1. Damit sinkt die Zahl der relevanten Automaten auf $5184 - 1728 = 3456$.

4.1.3 Das Akzeptanzverhalten

Automaten, bei denen alle oder keine Zustände akzeptierend sind, akzeptieren nur triviale Mengen. Also müssen sie nicht betrachtet werden. Deshalb fallen zwei der acht Mengen von akzeptierenden Zuständen weg. Die Anzahl der Automaten sinkt also auf $6/8 * 3456 = 2592$.

Zu jedem Automaten existiert einer mit konträrem Akzeptanzverhalten. Wir müssen also nur eine Hälfte der Automaten betrachten. Zu jeder entstehende Komplexitätsklasse existiert dann noch die entsprechende co-Klasse. Die Akzeptierungsmengen {B}, {C} und {B,C} werden gestrichen. übrig bleiben also noch Automaten mit den Kodierungen AAxxxxxx, ABxxxxxx und ACxxxxxx. Damit sinkt die Anzahl der Automaten um die Hälfte auf $2592/2 = 1296$.

4.1.4 Nichtminimale Automaten

Ein Automat mit drei Zuständen ist nichtminimal, wenn zwei Zustände sich „ähnlich“ verhalten. Das heißt:

- Beide Zustände sind akzeptierend oder beide Zustände sind nicht akzeptierend.
- Wird in beiden Zuständen dasselbe Zeichen gelesen, so gilt: Beide Zustände gehen in einen der beiden Zustände über oder beide Zustände gehen in den dritten Zustand über.

Die Kodierungen aller nichtminimalen Automaten, bei denen die Zustände A und B akzeptierend sind, lauten:

ABACACxx*
ABACBCxx*
ABBCACxx
ABBCBCxx
ABCACAxx*
ABCACBxx*
ABCBCAxx
ABCBCBxx
ABCCCCxx*

Also müßten nun $3^2 * 9 = 81$ Automaten gestrichen werden. Aber diese Streichmenge überschneidet sich mit der Menge der Automaten, die nur zwei Zustände benutzen. Alle Kodierungen, die mit einem ‚*‘gekennzeichnet sind, stellen Automaten mit zwei benutzten Zuständen dar, wenn man xx durch AA, AC, CA oder CC ersetzt. Also können in den Zeilen mit Stern statt der 9 Automaten nur noch 5 gestrichen werden. Insgesamt können somit noch $4 * 9 + 5 * 5 = 61$ Automaten gestrichen werden.

Ähnlich verhält es sich auch bei den mit ACxxxxxx kodierten Automaten:

ACABxxAB*
ACABxxCB*
ACCBxxAB
ACCBxxCB
ACBAxxBA*
ACBAxxBC*
ACBCxxBA
ACBCxxBC
ACBBxxBB*

Hier treten die Überschneidungen bei folgenden Ersetzungen auf: AA, AB, BA und BB. Auch hier können wieder 61 Automaten gestrichen werden.

Bei Automaten, deren einziger akzeptierender Zustand A ist, führen folgende Kodierungen zu Nichtminimalität:

AAxxAAAA+	AAxxABAB*
AAxxABAC+	AAxxACAB-
AAxxACAC**	AAxxBABA*
AAxxCABA-	AAxxBACA+
AAxxCACA**	AAxxBBBB*
AAxxBBBC*	AAxxBBCB*
AAxxBBCC+	AAxxBCBB-
AAxxBCBC-	AAxxBCCB-
AAxxBCCC**	AAxxCBBB-
AAxxCBBC-	AAxxCBCB-
AAxxCBCC**	AAxxCCBB-
AAxxCCBC-	AAxxCCCB-
AAxxCCCC**	

Die Kennzeichnungen am Ende der Kodierung bedeuten folgendes:

* Es gibt Überschneidungen, wenn man xx durch AA, AB, BA oder BB ersetzt.

** Es gibt Überschneidungen, wenn man xx durch AA, AC, CA oder CC ersetzt.

+ Es gibt Überschneidungen, wenn man xx durch AA, AB, AC, BA, BB, CA oder CC ersetzt.

- Es gibt Überschneidungen, wenn man xx durch AA ersetzt.

Hier können also $5*5+5*5+4*2+11*8 = 146$ Automaten gestrichen werden.

Am Ende verringert sich die Anzahl auf $1296 - (61+61+146) = 1028$.

4.1.5 Spiegelsymmetrische Automaten

Zu jedem Automaten existiert ein „spiegelsymmetrischer“ Automat, bei dem die Rollen von Zustand B und C vertauscht sind. Da beide dieselbe Sprache akzeptieren, genügt es, einen von beiden zu betrachten.

Das Spiegelbild eines jeden der letzten 1028 Automaten befindet sich ebenfalls in der Menge der noch übrigen 1028: Wäre das Spiegelbild als nichtminimaler Automat gestrichen worden, so wäre auch der Originalautomat nichtminimal und somit gestrichen. Wäre das Spiegelbild gestrichen worden, weil es die Akzeptanzmengen {B},{C} oder {B,C} besäße, so hätte der Originalautomat die Akzeptanzmengen {C}, {B} oder {B,C} und wäre damit auch gestrichen.

Bei den letzten 1028 Automaten ist auch jeder Automat von seinem Spiegelbild verschieden; denn eine notwendige Voraussetzung dafür, daß ein Automat seinem Spiegelbild gleicht, ist die Kodierung xxAAxxxx, und solche Automaten wurden zu Beginn ausgeschlossen.

Werden also alle Spiegelbilder gestrichen, so halbiert sich die Anzahl der Automaten. Es verbleiben $1028/2 = 514$.

4.1.6 Die Vertauschung von 0 und 1

Es wurde schon erwähnt, daß zwei Blattsprachen, bei denen 0 und 1 invertiert sind, dieselbe Komplexitätsklasse ergeben. Wird die Originalsprache von einem Automaten erkannt, so wird die invertierte Sprache von einem Automaten erkannt, bei dem in der Überführungsfunktion die Rollen von 0 und 1 vertauscht sind. Wir nennen einen solchen Automaten ebenfalls invertiert.

Nun ist zwar bei den letzten 514 Exemplaren nicht zu jedem Automaten der invertierte vorhanden, aber wenn das nicht der Fall ist so existiert doch der invertierte des spiegelsymmetrischen Automaten. So läßt sich wieder zu jedem Automaten das Pendant finden. Problematisch sind nur die Fälle, bei denen der Automat seinem Pendant entspricht:

- Der Automat entspricht dem invertierten Automaten. Das kann nur geschehen, falls der Automat folgende Kodierung hat: $x_1x_2x_3x_3x_4x_4$-x_5x_5. Nachdem man alle vorher gestrichenen Automaten entfernt hat, bleiben folgende 6 Fälle übrig:

  AABBCCAA
ABBBCCAA
ACBBCCAA
ABBBCCBB
ABBBCCCC
ACBBCCCC

- Der Automat entspricht dem spiegelsymmetrischen invertierten Automaten. Das kann nur geschehen, falls der Automat die Kodierung

AA$x_1\,\overline{x_1}\,x_2\,x_3\,\overline{x_3}\,\overline{x_2}$ besitzt, wobei $\overline{\mathrm{A}} = \mathrm{A}$, $\overline{\mathrm{B}} = \mathrm{C}$ und $\overline{\mathrm{C}} = \mathrm{B}$ gilt. Es bleiben schließlich 4 Automaten dieser Form übrig:

AABCABCA
AABCACBA
AABCBAAC
AABCCAAB

Man kann also von $514 - 10 = 504$ Automaten die Hälfte streichen und es bleiben schließlich $504/2 + 10 = 262$ Automaten übrig.

4.2 Alle relevanten Blattsprachen

Mit Hilfe eines Computerprogrammes konnten zu allen übriggebliebenen Automaten die entsprechenden regulären Mengen ermittelt werden. Dabei wurden Ausdrücke der Form ‚$(0 \vee 1)$' durch ‚x' ersetzt.

n	Automat n	S_n
1	AA:AB.AC.AA	(0*1(0 ∨ 1x))*0*
2	AA:AB.AC.BA	(0*1(0 ∨ 1(01)*(00 ∨ 1)))*0*
3	AA:AB.AC.BB	(0*1(1x)*0)*0*
4	AA:AB.AC.BC	(0*1(11*0)*0)*0*
5	AA:AB.AC.CA	(0*1(0 ∨ 10*1))*0*
6	AA:AB.AC.CB	(0*1(10*1)*0)*0*
7	AA:AB.AC.CC	(0*10)*0*
8	AA:AB.BC.AA	0* ∨ 0*10*1(x0*10*1)*x0*
9	AA:AB.BC.AB	0* ∨ 0*10*1((10*1)*00*10*1)*(10*1)*00*
10	AA:AB.BC.AC	0* ∨ 0*10*1(1*00*10*1)*1*00*
11	AA:AB.BC.BA	0* ∨ 0*10*1((00*1)*10*10*1)*(00*1)*10*
12	AA:AB.BC.CA	0* ∨ 0*10*1(0*10*10*1)*0*10*
13	AA:AB.CA.AA	(0*1(0x ∨ 1))*0*
14	AA:AB.CA.AB	(0*1(0(10)*(0 ∨ 11) ∨ 1))*0*
15	AA:AB.CA.AC	(0*1(01*0 ∨ 1))*0*
16	AA:AB.CA.BB	(0*1(0x)*1)*0*
17	AA:AB.CA.BC	(0*1(01*0)*1)*0*
18	AA:AB.CA.CB	(0*1(00*1)*1)*0*
19	AA:AB.CA.CC	(0*11)*0*
20	AA:AB.CB.AA	0* ∨ 0*11*0(x0*11*0)*x0*
21	AA:AB.CB.AB	0* ∨ 0*11*0((11*0)*00*11*0)*(11*0)*00*
22	AA:AB.CB.AC	0* ∨ 0*11*0(1*00*11*0)*1*00*
23	AA:AB.CB.BA	0* ∨ 0*11*0((01*0)*10*11*0)*(01*0)*10*
24	AA:AB.CB.CA	0* ∨ 0*11*0(0*10*11*0)*0*10*
25	AA:AB.CC.AA	0* ∨ 0*1x(x0*1x)*x0*
26	AA:AB.CC.AB	0* ∨ 0*1x((1x)*00*1x)*(1x)*00*
27	AA:AB.CC.AC	0* ∨ 0*1x(1*00*1x)*1*00*
28	AA:AB.CC.BA	0* ∨ 0*1x((0x)*10*1x)*(0x)*10*
29	AA:AB.CC.CA	0* ∨ 0*1x(0*10*1x)*0*10*
30	AA:BB.AC.AA	(x(0 ∨ 1x))*
31	AA:BB.AC.BA	(x(0 ∨ 1(01)*(00 ∨ 1)))*
32	AA:BB.AC.BB	(x(1x)*0)*
33	AA:BB.AC.BC	(x(11*0)*0)*
34	AA:BB.AC.CA	(x(0 ∨ 10*1))*
35	AA:BB.AC.CB	(x(10*1)*0)*
36	AA:BB.AC.CC	(x0)*
37	AA:BB.BC.AA	x0*1(xx0*1)*x ∨ ε
38	AA:BB.BC.AB	x0*1((10*1)*0x0*1)*(10*1)*0 ∨ ε
39	AA:BB.BC.AC	x0*1(1*0x0*1)*1*0 ∨ ε
40	AA:BB.BC.BA	x0*1((00*1)*1x0*1)*(00*1)*1 ∨ ε
41	AA:BB.BC.CA	x0*1(0*1x0*1)*0*1 ∨ ε
42	AA:BB.CC.AA	xx(xxx)*x ∨ ε
43	AA:BB.CC.AB	xx((1x)*0xx)*(1x)*0 ∨ ε

44	AA:BB.CC.AC	xx(1*0xx)*1*0 ∨ ε
45	AA:BC.AA.AB	((1(0 ∨ 1x))*0x)*(1(0 ∨ 1x))*
46	AA:BC.AA.AC	((11*0)*0x)*(11*0)*
47	AA:BC.AA.BA	((1(0x ∨ 1))*0x)*(1(0x ∨ 1))*
48	AA:BC.AA.BB	((0 ∨ 1x)x)*
49	AA:BC.AA.BC	((0 ∨ 11*0)x)*
50	AA:BC.AA.CA	((10*1)*0x)*(10*1)*
51	AA:BC.AA.CB	((0 ∨ 10*1)x)*
52	AA:BC.AA.CC	(0x)*
53	AA:BC.AB.BA	((1(01*0 ∨ 1))*01*0)*(1(01*0 ∨ 1))*
54	AA:BC.AB.BB	((0 ∨ 1x)1*0)*
55	AA:BC.AB.BC	((0 ∨ 11*0)1*0)*
56	AA:BC.AB.CA	((10*1)*01*0)*(10*1)*
57	AA:BC.AB.CB	((0 ∨ 10*1)1*0)*
58	AA:BC.AB.CC	(01*0)*
59	AA:BC.AC.BA	((1(01)*(00 ∨ 1))*0(0 ∨ 1(01)*(00 ∨ 1)))*(1(01)*-(00 ∨ 1))*
60	AA:BC.AC.BB	((0 ∨ 1x)(1x)*0)*
61	AA:BC.AC.BC	((0 ∨ 11*0)(11*0)*0)*
62	AA:BC.AC.CB	((0 ∨ 10*1)(10*1)*0)*
63	AA:BC.AC.CC	(00)*
64	AA:BC.BA.AB	((1(0 ∨ 10*1))*00*1)*(1(0 ∨ 10*1))*
65	AA:BC.BA.AC	((11*0)*00*1)*(11*0)*
66	AA:BC.BA.BB	((0 ∨ 1x)0*1)*
67	AA:BC.BA.BC	((0 ∨ 11*0)0*1)*
68	AA:BC.BA.CB	((0 ∨ 10*1)0*1)*
69	AA:BC.BA.CC	(00*1)*
70	AA:BC.BB.AB	(10)*
71	AA:BC.BC.AB	(1(10*1)*0)* ∨ (1(10*1)*0)*00*1((10*1)*0(1(10*-1)*0)*00*1)*(10*1)*0(1(10*1)*0)*
72	AA:BC.CA.AB	((1(10)*(0 ∨ 11))*0(0(10)*(0 ∨ 11) ∨ 1))*(1(10)*(0 ∨ 11))*
73	AA:BC.CA.BB	((0 ∨ 1x)(0x)*1)*
74	AA:BC.CA.CB	((0 ∨ 10*1)(00*1)*1)*
75	AB:AB.AC.AA	(0*1(0 ∨ 1x))*(0* ∨ 0*1)
76	AB:AB.AC.AB	(0*1(0 ∨ 1(11)*(0 ∨ 10)))*(0* ∨ 0*1 ∨ 0*11(11)*1)
77	AB:AB.AC.AC	(0*1(0 ∨ 11*0))*(0* ∨ 0*1)
78	AB:AB.AC.BA	(0*1(0 ∨ 1(01)*(00 ∨ 1)))*(0* ∨ 0*1 ∨ 0*11(01)*0)
79	AB:AB.AC.BB	(0*1(1x)*0)*(0* ∨ 0*1(1x)*)
80	AB:AB.AC.BC	(0*1(11*0)*0)*(0* ∨ 0*1(11*0)*)
81	AB:AB.AC.CA	(0*1(0 ∨ 10*1))*(0* ∨ 0*1)
82	AB:AB.AC.CB	(0*1(10*1)*0)*(0* ∨ 0*1(10*1)*)
83	AB:AB.AC.CC	(0*10)*(0* ∨ 0*1)
84	AB:AB.BC.AA	0* ∨ 0*1(0* ∨ 0*1(x0*10*1)*(x0*10* ∨ x0*))

85	AB:AB.BC.AB	0* ∨ 0*1(0* ∨ 0*1((10*1)*00*10*1)*((10*1)*10* ∨ (10*1)*00*10* ∨ (10*1)*00*))
86	AB:AB.BC.AC	0* ∨ 0*1(0* ∨ 0*1(1*00*10*1)*(1*00*10* ∨ 1*00*-))
87	AB:AB.BC.BA	0* ∨ 0*1(0* ∨ 0*1((00*1)*10*10*1)*((00*1)*10*-10* ∨ (00*1)*10* ∨ (00*1)*00*))
88	AB:AB.BC.BB	0* ∨ 0*1(0* ∨ 0*1(x0*1)*x0*)
89	AB:AB.BC.BC	0* ∨ 0*1(0* ∨ 0*1(1*00*1)*1*00*)
90	AB:AB.BC.CA	0* ∨ 0*1(0* ∨ 0*1(0*10*10*1)*(0*10*10* ∨ 0*10*-))
91	AB:AB.BC.CB	0* ∨ 0*1(0* ∨ 0*1(0*10*1)*0*10*)
92	AB:AB.BC.CC	0* ∨ 0*10*
93	AB:AB.CA.AA	(0*1(0x ∨ 1))*(0* ∨ 0*1)
94	AB:AB.CA.AB	(0*1(0(10)*(0 ∨ 11) ∨ 1))*(0* ∨ 0*1 ∨ 0*10(10)*1)
95	AB:AB.CA.AC	(0*1(01*0 ∨ 1))*(0* ∨ 0*1)
96	AB:AB.CA.BA	(0*1(0(00)*(01 ∨ 1) ∨ 1))*(0* ∨ 0*1 ∨ 0*10(00)*0)
97	AB:AB.CA.BB	(0*1(0x)*1)*(0* ∨ 0*1(0x)*)
98	AB:AB.CA.BC	(0*1(01*0)*1)*(0* ∨ 0*1(01*0)*)
99	AB:AB.CA.CA	(0*1(00*1 ∨ 1))*(0* ∨ 0*1)
100	AB:AB.CA.CB	(0*1(00*1)*1)*(0* ∨ 0*1(00*1)*)
101	AB:AB.CA.CC	(0*11)*(0* ∨ 0*1)
102	AB:AB.CB.AA	0* ∨ 0*1(1* ∨ 1*0(x0*11*0)*(x0*11* ∨ x0*))
103	AB:AB.CB.AB	0* ∨ 0*1(1* ∨ 1*0((11*0)*00*11*0)*((11*0)*11* ∨ (11*0)*00*11* ∨ (11*0)*00*))
104	AB:AB.CB.AC	0* ∨ 0*1(1* ∨ 1*0(1*00*11*0)*(1*00*11* ∨ 1*00*-))
105	AB:AB.CB.BA	0* ∨ 0*1(1* ∨ 1*0((01*0)*10*11*0)*((01*0)*10*-11* ∨ (01*0)*10* ∨ (01*0)*01*))
106	AB:AB.CB.BB	0* ∨ 0*1(1* ∨ 1*0(x1*0)*x1*)
107	AB:AB.CB.BC	0* ∨ 0*1(1* ∨ 1*0(1*01*0)*1*01*)
108	AB:AB.CB.CA	0* ∨ 0*1(1* ∨ 1*0(0*10*11*0)*(0*10*11* ∨ 0*10*-))
109	AB:AB.CB.CB	0* ∨ 0*1(1* ∨ 1*0(0*11*0)*0*11*)
110	AB:AB.CB.CC	0* ∨ 0*11*
111	AB:AB.CC.AA	0* ∨ 0*1(x(x0*1x)*(x0*1 ∨ x0*) ∨ ε)
112	AB:AB.CC.AB	0* ∨ 0*1(x((1x)*00*1x)*((1x)*1 ∨ (1x)*00*1 ∨ (1x)*00*) ∨ ε)
113	AB:AB.CC.AC	0* ∨ 0*1(x(1*00*1x)*(1*00*1 ∨ 1*00*) ∨ ε)
114	AB:AB.CC.BA	0* ∨ 0*1(x((0x)*10*1x)*((0x)*10*1 ∨ (0x)*10* ∨ (0x)*0) ∨ ε)
115	AB:AB.CC.BB	0* ∨ 0*1(x(xx)*x ∨ ε)
116	AB:AB.CC.BC	0* ∨ 0*1(x(1*0x)*1*0 ∨ ε)
117	AB:AB.CC.CA	0* ∨ 0*1(x(0*10*1x)*(0*10*1 ∨ 0*10*) ∨ ε)
118	AB:AB.CC.CB	0* ∨ 0*1(x(0*1x)*0*1 ∨ ε)
119	AB:AB.CC.CC	0* ∨ 0*1

120	AB:BB.AC.AA	(x(0 ∨ 1x))*(x ∨ ε)
121	AB:BB.AC.AB	(x(0 ∨ 1(11)*(0 ∨ 10)))*(x ∨ x1(11)*1 ∨ ε)
122	AB:BB.AC.AC	(x(0 ∨ 11*0))*(x ∨ ε)
123	AB:BB.AC.BA	(x(0 ∨ 1(01)*(00 ∨ 1)))*(x ∨ x1(01)*0 ∨ ε)
124	AB:BB.AC.BB	(x(1x)*0)*(x(1x)* ∨ ε)
125	AB:BB.AC.BC	(x(11*0)*0)*(x(11*0)* ∨ ε)
126	AB:BB.AC.CA	(x(0 ∨ 10*1))*(x ∨ ε)
127	AB:BB.AC.CB	(x(10*1)*0)*(x(10*1)* ∨ ε)
128	AB:BB.AC.CC	(x0)*(x ∨ ε)
129	AB:BB.BC.AA	x(0* ∨ 0*1(xx0*1)*(x ∨ xx0*)) ∨ ε
130	AB:BB.BC.AB	x(0* ∨ 0*1((10*1)*0x0*1)*((10*1)*10* ∨ (10*1)*0 ∨ (10*1)*0x0*)) ∨ ε
131	AB:BB.BC.AC	x(0* ∨ 0*1(1*0x0*1)*(1*0 ∨ 1*0x0*)) ∨ ε
132	AB:BB.BC.BA	x(0* ∨ 0*1((00*1)*1x0*1)*((00*1)*1 ∨ (00*1)*-1x0* ∨ (00*1)*00*)) ∨ ε
133	AB:BB.BC.BB	x(0* ∨ 0*1(x0*1)*x0*) ∨ ε
134	AB:BB.BC.BC	x(0* ∨ 0*1(1*00*1)*1*00*) ∨ ε
135	AB:BB.BC.CA	x(0* ∨ 0*1(0*1x0*1)*(0*1 ∨ 0*1x0*)) ∨ ε
136	AB:BB.BC.CB	x(0* ∨ 0*1(0*10*1)*0*10*) ∨ ε
137	AB:BB.BC.CC	x0* ∨ ε
138	AB:BB.CC.AA	x(x(xxx)*(x ∨ xx) ∨ ε) ∨ ε
139	AB:BB.CC.AB	x(x((1x)*0xx)*((1x)*1 ∨ (1x)*0 ∨ (1x)*0x) ∨ ε) ∨ ε
140	AB:BB.CC.AC	x(x(1*0xx)*(1*0 ∨ 1*0x) ∨ ε) ∨ ε
141	AB:BB.CC.BB	x(x(xx)*x ∨ ε) ∨ ε
142	AB:BB.CC.BC	x(x(1*0x)*1*0 ∨ ε) ∨ ε
143	AB:BB.CC.CC	x ∨ ε
144	AB:BC.AA.AA	((1x)*0x)*((1x)* ∨ (1x)*0)
145	AB:BC.AA.AB	((1(0 ∨ 1x))*0x)*((1(0 ∨ 1x))* ∨ (1(0 ∨ 1x))*11 ∨ (1(0 ∨ 1x))*0)
146	AB:BC.AA.AC	((11*0)*0x)*((11*0)* ∨ (11*0)*0)
147	AB:BC.AA.BA	((1(0x ∨ 1))*0x)*((1(0x ∨ 1))* ∨ (1(0x ∨ 1))*10 ∨ (1(0x ∨ 1))*0)
148	AB:BC.AA.BB	((0 ∨ 1x)x)*((0 ∨ 1x) ∨ ε)
149	AB:BC.AA.BC	((0 ∨ 11*0)x)*((0 ∨ 11*0) ∨ ε)
150	AB:BC.AA.CA	((10*1)*0x)*((10*1)* ∨ (10*1)*0)
151	AB:BC.AA.CB	((0 ∨ 10*1)x)*((0 ∨ 10*1) ∨ ε)
152	AB:BC.AA.CC	(0x)*(0 ∨ ε)
153	AB:BC.AB.AA	((1x)*01*0)*((1x)* ∨ (1x)*01*)
154	AB:BC.AB.AB	((1(0 ∨ 11*0))*01*0)*((1(0 ∨ 11*0))* ∨ (1(0 ∨ 11*-0))*111* ∨ (1(0 ∨ 11*0))*01*)
155	AB:BC.AB.AC	((11*0)*01*0)*((11*0)* ∨ (11*0)*01*)
156	AB:BC.AB.BA	((1(01*0 ∨ 1))*01*0)*((1(01*0 ∨ 1))* ∨ (1(01*0 ∨ 1))*101* ∨ (1(01*0 ∨ 1))*01*)
157	AB:BC.AB.BB	((0 ∨ 1x)1*0)*((0 ∨ 1x)1* ∨ ε)

158	AB:BC.AB.BC	((0 ∨ 11*0)1*0)*((0 ∨ 11*0)1* ∨ ε)
159	AB:BC.AB.CA	((10*1)*01*0)*((10*1)* ∨ (10*1)*01*)
160	AB:BC.AB.CB	((0 ∨ 10*1)1*0)*((0 ∨ 10*1)1* ∨ ε)
161	AB:BC.AB.CC	(01*0)*(01* ∨ ε)
162	AB:BC.BA.AA	((1x)*00*1)*((1x)* ∨ (1x)*00*)
163	AB:BC.BA.AB	((1(0 ∨ 10*1))*00*1)*((1(0 ∨ 10*1))* ∨ (1(0 ∨ 10*1))*110* ∨ (1(0 ∨ 10*1))*00*)
164	AB:BC.BA.AC	((11*0)*00*1)*((11*0)* ∨ (11*0)*00*)
165	AB:BC.BA.BA	((1(00*1 ∨ 1))*00*1)*((1(00*1 ∨ 1))* ∨ (1(00*1 ∨ 1))*100* ∨ (1(00*1 ∨ 1))*00*)
166	AB:BC.BA.BB	((0 ∨ 1x)0*1)*((0 ∨ 1x)0* ∨ ε)
167	AB:BC.BA.BC	((0 ∨ 11*0)0*1)*((0 ∨ 11*0)0* ∨ ε)
168	AB:BC.BA.CA	((10*1)*00*1)*((10*1)* ∨ (10*1)*00*)
169	AB:BC.BA.CB	((0 ∨ 10*1)0*1)*((0 ∨ 10*1)0* ∨ ε)
170	AB:BC.BA.CC	(00*1)*(00* ∨ ε)
171	AB:BC.BB.AA	(1x)* ∨ (1x)*0x*
172	AB:BC.BB.AB	(10)* ∨ (10)*11x* ∨ (10)*0x*
173	AB:BC.BB.AC	(11*0)* ∨ (11*0)*0x*
174	AB:BC.BB.BA	(11)* ∨ (11)*10x* ∨ (11)*0x*
175	AB:BC.BB.BB	0x* ∨ 1xx* ∨ ε
176	AB:BC.BB.BC	0x* ∨ 11*0x* ∨ ε
177	AB:BC.BB.CA	(10*1)* ∨ (10*1)*0x*
178	AB:BC.BB.CB	0x* ∨ 10*1x* ∨ ε
179	AB:BC.BB.CC	0x* ∨ ε
180	AB:BC.CA.AA	((1x)*0(0x ∨ 1))*((1x)* ∨ (1x)*0)
181	AB:BC.CA.AB	((1(10)*(0 ∨ 11))*0(0(10)*(0 ∨ 11) ∨ 1))*((1(10)*(0 ∨ 11))* ∨ (1(10)*(0 ∨ 11))*1(10)*1 ∨ (1(10)*(0 ∨ 11))*0 ∨ (1(10)*(0 ∨ 11))*00(10)*1)
182	AB:BC.CA.AC	((11*0)*0(01*0 ∨ 1))*((11*0)* ∨ (11*0)*0)
183	AB:BC.CA.BA	((1(00)*(01 ∨ 1))*0(0(00)*(01 ∨ 1) ∨ 1))*((1(00)*(01 ∨ 1))* ∨ (1(00)*(01 ∨ 1))*1(00)*0 ∨ (1(00)*(01 ∨ 1))*0 ∨ (1(00)*(01 ∨ 1))*00(00)*0)
184	AB:BC.CA.BB	((0 ∨ 1x)(0x)*1)*((0 ∨ 1x)(0x)* ∨ ε)
185	AB:BC.CA.BC	((0 ∨ 11*0)(01*0)*1)*((0 ∨ 11*0)(01*0)* ∨ ε)
186	AB:BC.CA.CA	((10*1)*0(00*1 ∨ 1))*((10*1)* ∨ (10*1)*0)
187	AB:BC.CA.CB	((0 ∨ 10*1)(00*1)*1)*((0 ∨ 10*1)(00*1)* ∨ ε)
188	AB:BC.CA.CC	(01)*(0 ∨ ε)
189	AB:BC.CB.AA	(1x)* ∨ (1x)*0(1* ∨ 1*0(x(1x)*01*0)*(x(1x)*01* ∨ x(1x)*))
190	AB:BC.CB.AB	(1(11*0)*0)* ∨ (1(11*0)*0)*1(11*0)*11* ∨ (1(11*0)*0)*0(1* ∨ 1*0((11*0)*0(1(11*0)*0)*01*0)*((11*0)*11* ∨ (11*0)*0(1(11*0)*0)*01* ∨ (11*0)*0(1(11*0)*0)*1(11*0)*11* ∨ (11*0)*0(1(11*0)*0)*))

191	AB:BC.CB.AC	(11*0)* ∨ (11*0)*0(1* ∨ 1*0(1*0(11*0)*01*0)*(1*-0(11*0)*01* ∨ 1*0(11*0)*))
192	AB:BC.CB.BA	(1(01*0)*1)* ∨ (1(01*0)*1)*1(01*0)*01* ∨ (1(01*0)*1)*0(1* ∨ 1*0((01*0)*1(1(01*0)*1)*01*-0)*((01*0)*1(1(01*0)*1)*01* ∨ (01*0)*1(1(01*-0)*1)*1(01*0)*01* ∨ (01*0)*1(1(01*0)*1)* ∨ (01*0)*01*))
193	AB:BC.CB.BB	0(1* ∨ 1*0(x1*0)*x1*) ∨ 1(x1*0)*x1* ∨ ε
194	AB:BC.CB.BC	0(1* ∨ 1*0(1*01*0)*1*01*) ∨ 1(1*01*0)*1*01* ∨ ε
195	AB:BC.CB.CA	(10*1)* ∨ (10*1)*0(1* ∨ 1*0(0*1(10*1)*01*0)*(0*-1(10*1)*01* ∨ 0*1(10*1)*))
196	AB:BC.CB.CB	0(1* ∨ 1*0(0*11*0)*0*11*) ∨ 1(0*11*0)*0*11* ∨ ε
197	AB:BC.CB.CC	01* ∨ ε
198	AB:BC.CC.AA	(1x)* ∨ (1x)*0(x(x(1x)*0x)*(x(1x)*0 ∨ x(1x)*) ∨ ε)
199	AB:BC.CC.AB	(1(1x)*0)* ∨ (1(1x)*0)*1(1x)*1 ∨ (1(1x)*0)*-0(x((1x)*0(1(1x)*0)*0x)*((1x)*1 ∨ (1x)*0(1(1x)*-0)*0 ∨ (1x)*0(1(1x)*0)*1(1x)*1 ∨ (1x)*0(1(1x)*-0)*) ∨ ε)
200	AB:BC.CC.AC	(11*0)* ∨ (11*0)*0(x(1*0(11*0)*0x)*(1*0(11*0)*0 ∨ 1*0(11*0)*) ∨ ε)
201	AB:BC.CC.BA	(1(0x)*1)* ∨ (1(0x)*1)*1(0x)*0 ∨ (1(0x)*1)*-0(x((0x)*1(1(0x)*1)*0x)*((0x)*1(1(0x)*1)*0 ∨ (0x)*1(1(0x)*1)*1(0x)*0 ∨ (0x)*1(1(0x)*1)* ∨ (0x)*0) ∨ ε)
202	AB:BC.CC.BB	0(x(xx)*x ∨ ε) ∨ 1(xx)*x ∨ ε
203	AB:BC.CC.BC	0(x(1*0x)*1*0 ∨ ε) ∨ 1(1*0x)*1*0 ∨ ε
204	AB:BC.CC.CA	(10*1)* ∨ (10*1)*0(x(0*1(10*1)*0x)*(0*1(10*1)*0 ∨ 0*1(10*1)*) ∨ ε)
205	AB:BC.CC.CB	0(x(0*1x)*0*1 ∨ ε) ∨ 1(0*1x)*0*1 ∨ ε
206	AB:BC.CC.CC	0 ∨ ε
207	AC:AB.AC.AA	(0*1(0 ∨ 1x))*(0* ∨ 0*11)
208	AC:AB.AC.AC	(0*1(0 ∨ 11*0))*(0* ∨ 0*111*)
209	AC:AB.AC.BA	(0*1(0 ∨ 1(01)*(00 ∨ 1)))*(0* ∨ 0*11(01)*)
210	AC:AB.AC.BB	(0*1(1x)*0)*(0* ∨ 0*1(1x)*1)
211	AC:AB.AC.BC	(0*1(11*0)*0)*(0* ∨ 0*1(11*0)*11*)
212	AC:AB.AC.CA	(0*1(0 ∨ 10*1))*(0* ∨ 0*110*)
213	AC:AB.AC.CC	(0*10)*(0* ∨ 0*11x*)
214	AC:AB.BC.AA	0* ∨ 0*10*1(x0*10*1)*(x0* ∨ ε)
215	AC:AB.BC.AC	0* ∨ 0*10*1(1*00*10*1)*(1* ∨ 1*00*)
216	AC:AB.BC.BA	0* ∨ 0*10*1((00*1)*10*10*1)*((00*1)* ∨ (00*1)*-10*)
217	AC:AB.BC.BB	0* ∨ 0*10*1(x0*1)*
218	AC:AB.BC.BC	0* ∨ 0*10*1(1*00*1)*1*
219	AC:AB.BC.CA	0* ∨ 0*10*1(0*10*10*1)*(0* ∨ 0*10*)

220	AC:AB.BC.CC	0* ∨ 0*10*1x*
221	AC:AB.CA.AA	(0*1(0x ∨ 1))*(0* ∨ 0*10)
222	AC:AB.CA.AC	(0*1(01*0 ∨ 1))*(0* ∨ 0*101*)
223	AC:AB.CA.BA	(0*1(0(00)*(01 ∨ 1) ∨ 1))*(0* ∨ 0*10(00)*)
224	AC:AB.CA.BB	(0*1(0x)*1)*(0* ∨ 0*1(0x)*0)
225	AC:AB.CA.BC	(0*1(01*0)*1)*(0* ∨ 0*1(01*0)*01*)
226	AC:AB.CA.CA	(0*1(00*1 ∨ 1))*(0* ∨ 0*100*)
227	AC:AB.CA.CC	(0*11)*(0* ∨ 0*10x*)
228	AC:AB.CB.AA	0* ∨ 0*11*0(x0*11*0)*(x0* ∨ ε)
229	AC:AB.CB.AC	0* ∨ 0*11*0(1*00*11*0)*(1* ∨ 1*00*)
230	AC:AB.CB.BA	0* ∨ 0*11*0((01*0)*10*11*0)*((01*0)* ∨ (01*0)*-10*)
231	AC:AB.CB.BB	0* ∨ 0*11*0(x1*0)*
232	AC:AB.CB.BC	0* ∨ 0*11*0(1*01*0)*1*
233	AC:AB.CB.CA	0* ∨ 0*11*0(0*10*11*0)*(0* ∨ 0*10*)
234	AC:AB.CB.CC	0* ∨ 0*11*0x*
235	AC:AB.CC.AA	0* ∨ 0*1x(x0*1x)*(x0* ∨ ε)
236	AC:AB.CC.AC	0* ∨ 0*1x(1*00*1x)*(1* ∨ 1*00*)
237	AC:AB.CC.BA	0* ∨ 0*1x((0x)*10*1x)*((0x)* ∨ (0x)*10*)
238	AC:AB.CC.BB	0* ∨ 0*1x(xx)*
239	AC:AB.CC.BC	0* ∨ 0*1x(1*0x)*1*
240	AC:AB.CC.CA	0* ∨ 0*1x(0*10*1x)*(0* ∨ 0*10*)
241	AC:AB.CC.CC	0* ∨ 0*1xx*
242	AC:BB.AC.AA	(x(0 ∨ 1x))*(x1 ∨ ε)
243	AC:BB.AC.AB	(x(0 ∨ 1(11)*(0 ∨ 10)))*(x1(11)* ∨ ε)
244	AC:BB.AC.AC	(x(0 ∨ 11*0))*(x11* ∨ ε)
245	AC:BB.AC.BA	(x(0 ∨ 1(01)*(00 ∨ 1)))*(x1(01)* ∨ ε)
246	AC:BB.AC.BC	(x(11*0)*0)*(x(11*0)*11* ∨ ε)
247	AC:BB.AC.CA	(x(0 ∨ 10*1))*(x10* ∨ ε)
248	AC:BB.AC.CB	(x(10*1)*0)*(x(10*1)*10* ∨ ε)
249	AC:BB.AC.CC	(x0)*(x1x* ∨ ε)
250	AC:BB.BC.AA	x0*1(xx0*1)*(x ∨ ε) ∨ ε
251	AC:BB.BC.AB	x0*1((10*1)*0x0*1)*((10*1)* ∨ (10*1)*0) ∨ ε
252	AC:BB.BC.AC	x0*1(1*0x0*1)*(1* ∨ 1*0) ∨ ε
253	AC:BB.BC.BA	x0*1((00*1)*1x0*1)*((00*1)* ∨ (00*1)*1) ∨ ε
254	AC:BB.BC.BC	x0*1(1*00*1)*1* ∨ ε
255	AC:BB.BC.CA	x0*1(0*1x0*1)*(0* ∨ 0*1) ∨ ε
256	AC:BB.BC.CB	x0*1(0*10*1)*0* ∨ ε
257	AC:BB.BC.CC	x0*1x* ∨ ε
258	AC:BB.CC.AA	xx(xxx)*(x ∨ ε) ∨ ε
259	AC:BB.CC.AB	xx((1x)*0xx)*((1x)* ∨ (1x)*0) ∨ ε
260	AC:BB.CC.AC	xx(1*0xx)*(1* ∨ 1*0) ∨ ε
261	AC:BB.CC.BC	xx(1*0x)*1* ∨ ε
262	AC:BB.CC.CC	xxx* ∨ ε

4.3 Definierbare Komplexitätsklassen

4.3.1 Klassen von endlichem Anzahltyp

Es ist nun möglich, die Klassen 1–NP und co1–NP zu charakterisieren:

$$\begin{aligned} \text{1–NP} &= \text{coLeaf}^{\text{P}}(S_{220}) \\ \text{co1–NP} &= \text{Leaf}^{\text{P}}(S_{220}) \end{aligned}$$

Des weiteren können {0,1}–P und co{0,1}–P charakterisiert werden:

$$\begin{aligned} \{0{,}1\}\text{–P} &= \text{Leaf}^{\text{P}}(S_{92}) \\ \text{co}\{0{,}1\}\text{–P} &= \text{coLeaf}^{\text{P}}(S_{92}) \end{aligned}$$

Da leicht einzusehen ist, daß {0,1}–P = coNP gilt, erhalten wir folgendes Ergebnis:

$$\text{coNP} = \text{Leaf}^{\text{P}}(S_{92})$$

Die Klassen {0}–P und co{0}–P entsprechen coNP bzw. NP und können schon mit Automaten der Größe zwei beschrieben werden.

Mehr Klassen des Typs S–P sind mit Hilfe von Automaten der Größe drei nicht zu beschreiben. Die Automaten werden dazu benutzt, die Anzahl der Einsen im Blattwort zu speichern. Schon um die Frage „befinden sich zwei Einsen im Blattwort?“ beantworten zu können, benötigt man vier verschiedene Zustände:

1. Es wurde noch keine Eins gelesen.
2. Es wurde genau eine Eins gelesen.
3. Es wurden genau zwei Einsen gelesen.
4. Es wurden mehr als zwei Einsen gelesen.

Dies ist mit Automaten mit maximal drei Zuständen schon nicht mehr möglich.

4.3.2 Mod-Klassen

Die Mod-Klassen stellen eine Erweiterung von $\oplus$P dar, denn es gilt: $\oplus\text{P} = \text{Mod}_2\text{P}$.

Die Klassen {0}(mod 2)–P und {1}(mod 2)–P entsprechen $\oplus$P und wurden schon mit Hilfe von Automaten der Größe zwei beschrieben. Wir können nun noch folgende Mod-Klassen beschreiben:

$$
\begin{array}{ll}
\mathrm{Mod_3P} = \{1{,}2\}(\mathrm{mod}\ 3)\text{–P} & = \mathrm{coLeaf^P}(S_{12}) \\
\mathrm{coMod_3P} = \{0\}(\mathrm{mod}\ 3)\text{–P} & = \quad \mathrm{Leaf^P}(S_{12}) \\
\{1\}(\mathrm{mod}\ 3)\text{–P} & = \mathrm{coLeaf^P}(S_{219}) \\
\mathrm{co}\{1\}(\mathrm{mod}\ 3)\text{–P} = \{0{,}2\}(\mathrm{mod}\ 3)\text{–P} & = \quad \mathrm{Leaf^P}(S_{219}) \\
\{2\}(\mathrm{mod}\ 3)\text{–P} & = \mathrm{coLeaf^P}(S_{90}) \\
\mathrm{co}\{2\}(\mathrm{mod}\ 3)\text{–P} = \{0{,}1\}(\mathrm{mod}\ 3)\text{–P} & = \quad \mathrm{Leaf^P}(S_{90})
\end{array}
$$

In [BGH90] wurde bewiesen, daß alle diese Mod-Klassen zusammenfallen; also kann das Ergebnis vereinfacht werden:

$$\mathrm{Leaf^P}(S_{12}) = \mathrm{Leaf^P}(S_{219}) = \mathrm{Leaf^P}(S_{90}) = \mathrm{Mod_3P}$$

Alle Automaten haben die gleiche Übergangsfunktion: xxABBCCA. Sie besitzen also eine „ringförmige“ Struktur. Beim Lesen einer 0 verharren sie im aktuellen Zustand und beim Lesen einer 1 springen sie einen Zustand weiter, bzw. von Zustand C nach Zustand A. So ist es ihnen möglich, die Anzahl der gelesenen Einsen modulo 3 zu speichern.

Da Automaten zum Speichern von Anzahlen modulo k mindestens k Zustände benötigen, ist es also nicht möglich, noch andere Mod-Klassen mit Automaten der Größe drei zu beschreiben.

Mit Automaten der Größe 2 konnte die Klasse $\mathrm{P^{NP:\oplus P}}$ charakterisiert werden – sozusagen als Erweiterung von $\oplus$P. In derselben Weise kann nun $\mathrm{Mod_3P}$ erweitert werden und man erhält folgenden Satz:

Satz 4.1 $\mathrm{Leaf^P}(S_1) = \mathrm{P^{NP:Mod_3P[1]}}$

Beweis: $S_1 = (0 \vee 1)^*0(111)^* \vee (111)^*$
In der Tabelle wird zwar $S_1 = (0^*1(0 \vee 1(0 \vee 1)))^*0^*$ angegeben, aber dies sind nur verschiedene reguläre Ausdrücke für die gleiche reguläre Menge. Der Beweis entspricht fast wörtlich dem Beweis für Satz 3.6:

1. $\mathrm{Leaf^P}(S_1) \subseteq \mathrm{P^{NP:Mod_3P[1]}}$:

   Sei $A \in \mathrm{Leaf^P}(S_1)$ mittels der TM M. M besitze den maximalen Verzweigungsgrad b. Wir benutzen nun ein NP-Orakel O_M mit

   $$O_M = \{(x,y) : x \in \mathbb{N} \text{ und } y \in \{0,\ldots,b-1\}^* \text{ und ein Pfad von } M \text{ bei der Abarbeitung von } x, \text{ der durch } y \text{ beschrieben wird, ist möglich und der Teilbaum über dem von } y \text{ beschriebenen Pfad enthält eine 1 }\}$$

   und ein $\mathrm{Mod_3P}$-Orakel O'_M mit

$O'_M = \{(x,y) : x \in \mathbb{N}$ und $y \in \{0,\dots,b-1\}^*$ und ein Pfad von M bei der Abarbeitung von x, der durch y beschrieben wird, ist möglich und der von y beschriebene Pfad hat maximale Verzweigungstiefe und der Teilbaum rechts dieses Pfades enthält eine durch drei teilbare Anzahl von Nullen $\}$

Wir sollten zuerst zeigen, daß O'_M überhaupt eine $\mathrm{Mod}_3\mathrm{P}$-Menge ist:

Eine NPTM M'' simuliert bei der Eingabe (x,y) die NPTM M, wobei alle Pfade, die links von dem durch y codierten Pfad liegen, nicht beschritten werden. So entspricht der Berechnungsbaum von M'' dem rechts von Pfad y liegenden Teilbaum von M.

Nun sucht M' die letzte 1 im Blattwort von M auf x mit Hilfe des Orakels O_M. Sie tastet sich dabei Schritt für Schritt durch den Berechnungsbaum von M und fragt bei jeder Verzweigung, in welchen Teilbäumen oberhalb des aktuellen Knotens sich Einsen an den Blättern befinden und verfolgt den rechtesten dieser Pfade. Hat M' so die letzte 1 gefunden, so zählt sie die Anzahl der Blätter rechts von dieser 1 – die ja alle Nullen sind – mit Hilfe einer Frage an O'_M. Ist die Anzahl durch drei teilbar, so akzeptiert M' die Eingabe x, sonst wird x abgelehnt.

2. $\mathrm{P}^{\mathrm{NP}:\mathrm{Mod}_3\mathrm{P}[1]} \subseteq \mathrm{Leaf}^{\mathrm{P}}(S_1)$:

   Sei $A \in \mathrm{P}^{\mathrm{NP}:\mathrm{Mod}_3\mathrm{P}[1]}$ mittels der deterministischen TM M, die das NP-Orakel O_M und das $\mathrm{Mod}_3\mathrm{P}$-Orakel O'_M benutzt. Diese Orakel beruhen auf der Auswertung der Berechnungsbäume von NPTMs. Wir nennen diese NPTMs N bzw. N'. Dabei machen wir uns zunutze, daß $\mathrm{Leaf}^{\mathrm{P}}(S_{42}) = \mathrm{Mod}_3\mathrm{P}$ gilt (mit $S_{42} = ((0 \vee 1)(0 \vee 1)(0 \vee 1))^*$). Bei N' kommt es also nur auf die Anzahl der Blätter des Berechnungsbaumes an, nicht auf die Anzahl der Einsen.

   Die NPTM M', die $A \in \mathrm{Leaf}^{\mathrm{P}}(S_1)$ beweisen soll, funktioniert folgendermaßen:

   Es wird solange M simuliert, bis die erste Frage „$y_1 \in O_M$?“ auftaucht. Nun wird N mit Eingabe y_1 simuliert. N spannt nun einen Berechnungsbaum auf. An den Blättern des Baumes kann N nun akzeptieren oder ablehnen. Akzeptiert N auf einem Pfad, so verzeigt sich M' in zwei Pfade. Auf dem linken Pfad wird akzeptiert und auf dem rechten Pfad wird N mit der Orakelantwort 1 weitersimuliert. Lehnt N aber ab, so verzweigt M' in drei Pfade und lehnt auf allen drei Pfaden ab.

Eine Ausnahme bildet nur der linkeste Pfad des Berechnungsbaumes von N. Hier wird auf jeden Fall in zwei Pfade verzweigt, auf dem linken Pfad akzeptiert und auf dem rechten Pfad mit der Antwort von N weitergerechnet.

So wird sichergestellt, daß nur der Pfad, auf dem die richtige Orakelantwort gegeben wurde, für die weitere Berechnung von Bedeutung ist:

- Ist $y_1 \in O_M$, so gibt es einen rechtesten Pfad von N, der akzeptiert. Auf diesem Pfad wird ja mit der richtigen Orakelantwort weitergerechnet und alle Pfade weiter links werden durch den links vorangestellten Pfad obsolet, da nur die Anzahl der Nullen rechts von der letzten 1 für die Akzeptierung von Bedeutung ist. Alle Pfade mit Nullen (außer dem linkesten) werden durch die dreifache Ablehnung aus der Wertung genommen – es zählt ja am Ende nur, ob die Anzahl der Nullen durch drei teibar ist und ein durch drei teilbarer Summand hat auf diese Frage keinen Einfluß.
- Ist aber $y_1 \notin O_M$, ist der einzige Pfad, auf dem die Berechnung weitergeführt wird, der linkeste. Auf ihm wird nun mit der richtigen Antwort 0 weitergerechnet.

Bei jeder weiteren Orakelfrage an O_M wird genauso verfahren.

Kommt schließlich die Frage an O'_M, so wird M mit beiden Antworten hintereinander weitersimuliert, und es wird überprüft, ob das Ergebnis von M überhaupt von dieser Frage abhängt.

- Kommt in beiden Fällen dasselbe Ergebnis heraus, so verzweigt sich M' ein letztes Mal in zwei Pfade. Auf dem linken wird akzeptiert und auf dem rechten wird das Ergebnis von M ausgegeben. Es gilt
  $x \in A \leftrightarrow$ Das Ergebnis von M ist 1.
  $\leftrightarrow$ Das entscheidende Teilwort des Blattwortes von M' endet mit einer 1.
  $\leftrightarrow$ Das Blattwort von M' liegt in $\mathrm{Leaf}^{\mathrm{P}}(S_1)$.
- Sind die Ergebnisse verschieden, so spaltet sich M' in zwei Pfade auf. Auf dem linken wird akzeptiert.

  Auf dem rechten Pfad wird N' simuliert und jeder Pfad wird mit einer Ablehnung abgeschlossen. OBdA seien die Orakelantworten von O'_M identisch mit den Ergebnissen von M. Dann gilt:

$$
\begin{aligned}
x \in A &\leftrightarrow M \text{ akzeptiert } x.\\
&\leftrightarrow \text{Die Orakelantwort von } O'_M \text{ war positiv.}\\
&\leftrightarrow \text{Der Berechnungsbaum vom } N' \text{ hat eine durch drei}\\
&\quad\ \text{teilbare Anzahl von Blättern.}\\
&\quad\ \text{(Da ja } O'_M \in \mathrm{Leaf}^{\mathrm{P}}(S_{42}) \text{ mittels der NPTM } N' \text{ gilt.)}\\
&\leftrightarrow \text{Der entscheidende Teilbaum von } M' \text{ endet mit einer}\\
&\quad\ \text{durch drei teilbaren Zahl von Nullen.}\\
&\leftrightarrow \text{Das Blattwort von } M' \text{ liegt in } \mathrm{Leaf}^{\mathrm{P}}(S_1).
\end{aligned}
$$

Also gilt $A \in \mathrm{Leaf}^{\mathrm{P}}(S_1)$ mittels M'. □

4.3.3 Eine obere Schranke

Es ist uns möglich, eine obere Schranke für $\mathrm{BalancedLeaf}^{\mathrm{P}}(K_3)$ zu anzugeben:

Satz 4.2 $\mathrm{BalancedLeaf}^{\mathrm{P}}(K_3) \subseteq \mathrm{MOD{-}PH}$

Beweis: Satz 5.1.2 aus [HLS$^+$93] besagt: Sei K eine Klasse von Sprachen, die alle ein auflösbares syntaktisches Monoid besitzen. Dann gilt:

$$\mathrm{BalancedLeaf}^{\mathrm{P}}(K) \subseteq \mathrm{MOD{-}PH}.$$

Es genügt also, zu beweisen, daß K_3 nur Sprachen mit auflösbarem syntaktischen Monoid enthält. Dazu verwenden wir Ergebnisse aus der Algebra:

Sei w ein Wort aus $\{0,1\}^*$ und D ein DEA mit der der Zustandsmenge Z und der Übergangsfunktion f. Dann erzeugt die Menge der Abbildungen $a_w : Z \to Z$ mit $a_w(z) = \overline{f}(z,w)$ unter der Verknüpfung Komposition $(f_x \cdot f_y = f_{xy})$ ein Monoid, das sogenannte Transformationsmonoid von D.

Dieses Transformationsmonoid hat für Automaten mit höchstens drei Zuständen höchtstens $3^3 = 27$ Elemente. Aus der Algebra ist bekannt, daß die kleinste nichtauflösbare Gruppe 60 Elemente besitzt. Demnach ist jedes Transformationsmonoid eines Automaten mit höchstens drei Zuständen auflösbar.

Eine Sprache S aus K_3 zeichnet sich gerade dadurch aus, daß sie von einem Automaten mit drei Zuständen erkannt werden kann. Dies bedeutet, daß ein minimaler Automat, der S akzeptiert, höchstens drei Zustände besitzen kann. Also ist das Transformationsmonoid eines Minimalautomaten für S auflösbar.

Das syntaktische Monoid von S ist isomorph zum Transformationsmonoid eines Minimalautomaten für S und damit ebenfalls auflösbar. Also besitzen alle Sprachen aus K_3 ein auflösbares syntaktischen Monoid. □

Literatur

[BCS92] D. P. Bovet, P. Crescenzi, and R. Silvestri. A uniform approach to define complexity classes. *Theoretical Computer Science*, 104:263–283, 1992.

[BG86] A. Blass and Y. Gurevich. Henkin quantifiers and complete problems. *Annals of Pure and Applied Logic*, 32:1–16, 1986.

[BGH90] R. Beigel, J. Gill, and U. Hertrampf. Counting classes: thresholds, parity, mods, and fewness. In *Proceedings 7th Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 49–57. Springer Verlag, 1990.

[CH89] J. Cai and L. Hemachandra. On the power of parity polynomial time. In *Procedings of the 6th Symposium on Theoretical Aspects of Computer Science*, volume 349 of *Lecture Notes in Computer Science*, pages 229–239, 1989.

[GW87] T. Gundermann and G. Wechsung. Counting classes of finite acceptance types. *Computers and Artificial Intelligence*, 6:395–409, 1987.

[HH90] L. A. Hemachandra and A. Hoene. On sets with efficient implicit membership tests. In *5th Structure in Complexity Theory*, pages 11–19. IEEE Computer Society Press, 1990.

[HLS+93] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. W. Wagner. On the power of polynomial time bit-reductions. In *Proceedings 8th Structure in Complexity Theory Conference*, pages 200–207, 1993.

[Nie97] G. Niemann. Blattsprachen. In H. Vollmer, editor, *Komplexitätstheorie: Maschinen und Operatoren*, pages 27–38. Cuvillier Verlag Göttingen, 1997.

[PZ83] C. H. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings 6th GI Conference on Theoretical Computer Science*, volume 145 of *Lecture Notes in Computer Science*, pages 269–275. Springer Verlag, 1983.

[Ver93] N. K. Vereshchagin. Relativizable and non-relativizable theorems in the polynomial theory of algorithms. *Izvestija Rossijskoj Akademii Nauk*, 57:51–90, 1993. In Russian.

[Wag90] K. W. Wagner. Bounded query classes. *SIAM Journal on Computation*, 19(5):833–846, 1990.

[WW85] K. W. Wagner and G. Wechsung. On the boolean closure of NP. In *Proceedings of the 5th Conference on Fundamentals of Computation Theory*, volume 199 of *Lecture Notes in Computer Science*, pages 485–493, 1985.