

**Komplexität einer frühen Fehlererkennung  
beim Parsen von deterministisch  
kontextfreien Sprachen**

HANS-MARTIN FÜSSEL

**Diplomarbeit**

26. Juni 1992

Fachbereich Informatik  
Johann-Wolfgang-Goethe-Universität Frankfurt am Main

Betreuer: PROF. DR. DETLEF WOTSCHKE

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Frankfurt am Main, den 26. Juni 1992

## Zusammenfassung

Es ist bekannt, daß der Verzicht auf die Correct-Prefix-Eigenschaft zu exponentiellen Einsparungen bei der Größe von DPDAs führen kann. In der vorliegenden Arbeit werden vor allem solche DPDAs betrachtet, deren Verhalten nach dem Lesen einer fehlerhaften Eingabe *zwischen* den Extremen „sofortiges Stoppen“ und „unbeschränktes Weiterarbeiten“ liegt. Die Verzögerung, mit der ein DPDA fehlerhafte Eingaben zurückweist, wird mit den drei Verzögerungsmaßen Eingabeverzögerung, Zeitverzögerung und Gesamt-Zeitverzögerung quantifiziert. Davon ausgehend werden für jede DCFL  $L$  drei Verzögerungsspektren definiert, die die Größe eines minimalen DPDA für  $L$  in Abhängigkeit von der erlaubten Verzögerung beschreiben.

Es werden notwendige und hinreichende Bedingungen dafür angegeben, daß eine Zahlenfolge näherungsweise dem Spektrum der Eingabeverzögerung einer DCFL entspricht, und für jede solche Zahlenfolge wird eine „passende“ Sprache konstruiert. Desweiteren werden notwendige Bedingungen für die Spektren der Zeitverzögerung und der Gesamt-Zeitverzögerung hergeleitet sowie Sprachfamilien mit besonders interessanten Verzögerungsspektren angegeben.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
<b>2</b>	<b>Definitionen</b>	<b>8</b>
2.1	Notationen . . . . .	8
2.2	Deterministische Kellerautomaten . . . . .	10
2.3	Verzögerungsspektren . . . . .	15
<b>3</b>	<b>Grundlegende Ergebnisse</b>	<b>23</b>
3.1	Größenmaße für deterministische Kellerautomaten . . . . .	23
3.2	Schranken für Verzögerungsspektren . . . . .	29
3.3	Berechenbarkeit von Verzögerungsspektren . . . . .	39
<b>4</b>	<b>Existenz von Verzögerungsspektren</b>	<b>45</b>
4.1	Einführende Ergebnisse und Definitionen . . . . .	45
4.2	Das Spektrum der Eingabeverzögerung . . . . .	49
4.3	Das Spektrum der Zeitverzögerung . . . . .	60
4.4	Das Spektrum der Gesamt-Zeitverzögerung . . . . .	76
<b>5</b>	<b>Schlußbemerkung</b>	<b>82</b>
	<b>Literaturverzeichnis</b>	<b>84</b>

# Verzeichnis der Sätze

Satz 1 . . . . .	25
Satz 2 . . . . .	36
Korollar 2.1 . . . . .	36
Korollar 2.2 . . . . .	37
Korollar 2.3 . . . . .	37
Korollar 2.4 . . . . .	37
Satz 3 . . . . .	37
Korollar 3.1 . . . . .	38
Korollar 3.2 . . . . .	38
Korollar 3.3 . . . . .	38
Korollar 3.4 . . . . .	38
Satz 4 . . . . .	39
Korollar 4.1 . . . . .	39
Satz 5 . . . . .	43
Satz 6 . . . . .	44
Korollar 6.1 . . . . .	44
Satz 7 . . . . .	49
Korollar 7.1 . . . . .	58
Korollar 7.2 . . . . .	59
Korollar 7.3 . . . . .	59
Satz 8 . . . . .	60
Satz 9 . . . . .	72
Satz 10 . . . . .	76
Satz 11 . . . . .	80

# Kapitel 1

## Einleitung

In der Informatik spielt die Konstruktion von Compilern sowohl theoretisch als auch praktisch eine große Rolle. Compiler dienen zur Übersetzung eines Eingabetextes von einem formalen System in ein anderes. Die erste Aufgabe bei diesem Übersetzungsprozeß besteht darin, zu überprüfen, ob die Eingabe syntaktisch korrekt ist, d. h., ob sie durch die Grammatik ableitbar ist, für die der Compiler konstruiert wurde, und gegebenenfalls eine Ableitung anzugeben. Derjenige Teil des Compilers, der diese Überprüfung vornimmt, wird als Parser bezeichnet. Die im Zusammenhang mit Compilern betrachteten Sprachen gehören der Klasse der deterministisch kontextfreien Sprachen (deterministic context-free language, DCFL) an.

Bei der Konstruktion eines Parsers werden verschiedene Ziele verfolgt, von denen die folgenden besonders wichtig sind:

1. Der Parser sollte möglichst klein sein.
2. Der Parser sollte möglichst schnell sein.
3. Der Parser sollte fehlerhafte Eingaben möglichst früh als solche erkennen.

Um bei einer Abwägung der konkurrierenden Ziele zu einem optimalen Kompromiß zu gelangen, ist eine möglichst umfassende Kenntnis des Zusammenhangs zwischen diesen Zielen erforderlich. In der vorliegenden Arbeit wird untersucht, inwiefern die Größe eines minimalen Parsers für eine DCFL  $L$ , also die Anzahl der Zeichen, die nötig sind, um

den Parser als Programm hinzuschreiben, davon abhängt, mit welcher Verzögerung dieser minimale inkorrekte Präfix von  $L$  zurückweist<sup>1</sup>. Von den zwei Aufgaben eines Parsers, dem Erkennen einer Sprache und dem Produzieren einer Ableitung, ist in dieser Arbeit nur die erste von Interesse, wobei besonders darauf geachtet wird, mit welcher Verzögerung fehlerhafte Eingaben zurückgewiesen werden. Es ist deshalb nicht erforderlich, daß ein Parser auf einer bestimmten Grammatik basiert, und anstatt von einem Parser für eine Grammatik  $G$  wird von einem Parser für eine Sprache  $L$  gesprochen. Als Automatenmodell für den Parser werden die deterministischen Kellerautomaten (deterministic push-down automaton, DPDA) zugrundegelegt.

Diese Arbeit ist in fünf Kapitel untergliedert, wobei mehrere Abschnitte ein Kapitel bilden. Kapitel 2 enthält die Definitionen der im weiteren Verlauf der Arbeit verwendeten Begriffe. In Abschnitt 2.1 werden eine Reihe von Notationen vereinbart, in Abschnitt 2.2 werden wichtige Begriffe zu DPDAs definiert, und in Abschnitt 2.3 wird das Konzept der Verzögerung von DPDAs eingeführt. Intuitiv gesprochen ist die Verzögerung eines DPDA ein Maß dafür, wieviele „überflüssige“ Übergänge der Automat auf einer Eingabe ausführt. Es erscheint sinnvoll, drei Verzögerungsmaße zu verwenden: Ein Parser für eine Sprache  $L$  arbeitet mit Eingabeverzögerung  $d$  für ein  $d \in \mathbb{N}_{0,\infty}$ <sup>2</sup>, falls er nach dem Lesen eines minimalen inkorrekten Präfixes von  $L$  höchstens  $d$  weitere Zeichen liest; er arbeitet mit Zeitverzögerung  $d$ , falls er nach dem Lesen eines minimalen inkorrekten Präfixes von  $L$  kein weiteres Zeichen liest und höchstens  $d$   $\varepsilon$ -Übergänge ausführt; er arbeitet mit Gesamt-Zeitverzögerung  $d$ , falls er nach dem Lesen eines minimalen inkorrekten Präfixes von  $L$  überhaupt keinen Übergang mehr macht und auf keiner (korrekten) Eingabe mehr als  $d$   $\varepsilon$ -Übergänge ausführt. Nach dieser Definition arbeitet jeder Parser mit Eingabeverzögerung  $\infty$ . Ein Parser wird als Correct-Prefix-Parser bezeichnet, falls er mit Eingabeverzögerung 0 arbeitet; er wird als Viable-Prefix-Parser bezeichnet, falls er mit Zeitverzögerung 0 arbeitet. Die Zeitverzögerung ist nur bei Correct-Prefix-Parsern von Interesse, und die Gesamt-Zeitverzögerung wird sogar nur bei Viable-Prefix-Parsern betrachtet.

---

<sup>1</sup>Ein Wort  $x$  ist ein minimales inkorrektes Präfix von  $L$ , falls sich  $x$  (bezüglich  $L$ ) nicht mehr korrekt fortsetzen läßt, wohl aber jedes echte Präfix von  $x$ .

<sup>2</sup> $\mathbb{N}_{0,\infty}$  ist eine verkürzte Schreibweise für die Menge  $\mathbb{N} \cup \{0, \infty\}$ .

Die Schwierigkeit, minimale inkorrekte Präfixe von  $L$  zu erkennen, kann grob abgeschätzt werden durch den Größenunterschied zwischen einem beliebigen minimalen Parser für  $L$  und einem minimalen Correct-Prefix-Parser oder Viable-Prefix-Parser für  $L$ . Es ist bekannt, daß jede DCFL durch eine LR(1)-Grammatik erzeugt werden kann, und daß der zugehörige kanonische LR(1)-Parser ein Viable-Prefix-Parser ist ([2, Section 5.2.], [13, Theorem 6.1] sowie [15]). Der kanonische LR(1)-Parser für eine typische höhere Programmiersprache hat jedoch den Nachteil, so groß zu sein, daß er nicht mit vertretbarem Aufwand implementiert werden kann, denn immerhin können LR( $k$ )-Parser exponentiell größer sein als die zugrundeliegenden Grammatiken ([6], zitiert nach [7]). Der Verzicht auf die Correct-Prefix-Eigenschaft kann bei bestimmten Klassen von Parsern zu exponentiellen Einsparungen führen [8, 17, 18]. Auch der Verzicht auf die Viable-Prefix-Eigenschaft unter Beibehaltung der Correct-Prefix-Eigenschaft wurde untersucht, ohne jedoch die mögliche Einsparung zu quantifizieren [1, 3, 4].

Das Ziel der vorliegenden Arbeit geht darüber hinaus, lediglich den Größenunterschied zwischen einem beliebigen minimalen Parser für eine Sprache  $L$  und z. B. einem minimalen Correct-Prefix-Parser für  $L$  zu betrachten; vielmehr soll auch untersucht werden, wie sich dieser Größenunterschied auf den Bereich von Eingabeverzögerung 0 bis Eingabeverzögerung  $\infty$  verteilt. Formal geschieht dies über Verzögerungsspektren für deterministisch kontextfreie Sprachen, deren Definition sich an die in [10] eingeführten Spektren für reguläre Sprachen anlehnt. Für eine DCFL  $L$  werden drei Verzögerungsspektren definiert, und zwar eines für jedes der drei Verzögerungsmaße. Ein Verzögerungsspektrum ist eine unendliche Zahlenfolge mit Endpunkt. Das  $i$ -te Folgenglied für  $i \in \mathbb{N}_0$  ist die Größe eines minimalen DPDA für  $L$ , der mit Verzögerung  $i$  (entsprechend dem jeweiligen Verzögerungsmaß) arbeitet, und das letzte Folgenglied ist die Größe eines minimalen DPDA für  $L$ , der mit Verzögerung  $\infty$  arbeitet. Ein Verzögerungsspektrum von  $L$  quantifiziert also die Schwierigkeit, minimale inkorrekte Präfixe von  $L$  früh zu erkennen, *unabhängig* von einem bestimmten Automaten.

Kapitel 3 enthält verschiedene grundlegende Ergebnisse. In Abschnitt 3.1 werden zwei Größenmaße für DPDAs, die Größe und die Produkt-Größe, miteinander verglichen. Es zeigt sich, daß die Verwendung der Größe und der Produkt-Größe nur dann zu vergleichbaren Ergebnissen führt, wenn die Länge desjenigen Kellerabschnitts, der in einem Übergang verändert werden kann, beschränkt wird.

In Abschnitt 3.2 werden obere und untere Schranken für Verzögerungsspektren hergeleitet. Zunächst wird eine Konstruktion angegeben, die zu einem beliebigen DPDA einen äquivalenten, exponentiell größeren DPDA angibt, der die Viable-Prefix-Eigenschaft besitzt. Daraus ergibt sich unmittelbar eine exponentielle obere Schranke für den Größenunterschied in Spektren der Eingabeverzögerung und der Zeitverzögerung<sup>3</sup>. Zur einfacheren Erklärung der folgenden Ergebnisse bezeichne  $\sigma_d^Z(L)$  die Größe eines minimalen DPDA, der die Sprache  $L$  mit Zeitverzögerung  $d$  für ein  $d \in \mathbb{N}_{0,\infty}$  erkennt. Für jede Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$  mit  $\sigma_0^Z(L_n) \geq n^n$  für alle  $n \in \mathbb{N}$  gilt  $\sigma_d^Z(L_n) = n^{\Omega(n/(d+1))}$ ; dasselbe Ergebnis ist auch für die Gesamt-Zeitverzögerung gültig. Für eine Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$  mit  $\sigma_0^Z(L_n) \geq n^n$  und  $\sigma_n^Z(L_n) = n^{\Theta(1)}$  für alle  $n \in \mathbb{N}$  läßt sich sogar das gesamte Spektrum der Zeitverzögerung von  $L_n$  näherungsweise angeben; für alle  $d \in \mathbb{N}_{0,\infty}$  gilt  $\sigma_d^Z(L_n) = n^{\Theta(1+n/(d+1))}$ . Weiterhin stellt sich heraus, daß  $L$  genau dann mit endlicher Gesamt-Zeitverzögerung erkannt werden kann, wenn  $L$  eine Realzeit-DCFL ist. Eine *globale* obere Schranke für den Größenunterschied im Spektrum der Gesamt-Zeitverzögerung einer Realzeit-DCFL konnte in dieser Arbeit nicht gezeigt werden. Da aus der Existenz einer solchen oberen Schranke die Existenz eines Algorithmus zur Lösung des bislang noch ungelösten Teilklassenproblems (DCFL, Realzeit-DCFL) folgen würde<sup>4</sup>, war ein einfacher Beweis hierfür jedoch auch nicht zu erwarten.

In Abschnitt 3.3 wird die Berechenbarkeit von Verzögerungsspektren untersucht. Da eine monoton fallende, beschränkte, diskrete Zahlenfolge auch dann nur einen endlichen Informationsgehalt hat, wenn sie selbst unendlich ist, spricht per se nichts gegen die effektive Berechenbarkeit von Verzögerungsspektren. Es zeigt sich, daß für jeden DPDA ermittelt werden kann, mit welcher Verzögerung dieser arbeitet. Zur Berechnung von Verzögerungsspektren ist allerdings noch ein Algorithmus vonnöten, der die Äquivalenz zweier DPDA entscheidet. Die Frage, ob ein solcher Algorithmus überhaupt existiert, gehört zu den bekanntesten offenen Problem aus der Theorie der Formalen Sprachen. Das Äquivalenz-

---

<sup>3</sup>Da die Kenntnis nur eines Verzögerungsspektrums nicht ausreicht, um von einem exponentiellen Größenunterschied zu sprechen, müßte man korrekter sagen, daß für eine Familie von Sprachen  $(L_n)_{n \in \mathbb{N}}$ , für die die minimalen Werte der Verzögerungsspektren linear in  $n$  wachsen, die maximalen Werte höchstens exponentiell in  $n$  wachsen können. Falls keine Mißverständnisse auftreten können, wird im folgenden jedoch weiterhin von einzelnen Verzögerungsspektren gesprochen, wo eigentlich unendliche Familien gemeint sind.

<sup>4</sup>Das Teilklassenproblem (DCFL, Realzeit-DCFL) besteht darin, für eine durch einen DPDA gegebene DCFL zu entscheiden, ob sie von einem Realzeit-DPDA akzeptiert wird.

problem für Realzeit-DPDAs ist jedoch gelöst, so daß zu einer Realzeit-DCFL alle drei Verzögerungsspektren berechenbar sind.

In Kapitel 4 wird die Existenz bestimmter Verzögerungsspektren gezeigt. Daraus folgt insbesondere, daß alle in Abschnitt 3.2 angegebenen Schranken für die Spektren der Eingabeverzögerung und der Zeitverzögerung annähernd, d. h., bis auf eine multiplikative Konstante im Exponenten, scharf sind. Da es sich durchweg um exponentielle Schranken handelt, ist diese „Unschärfe“ nicht gravierend. Das interessanteste Ergebnis dieser Arbeit wird in Abschnitt 4.2 gezeigt: Zu jeder Familie von monoton fallenden Funktionen  $(f_n)_{n \in \mathbb{N}}$  mit  $f_n : \mathbb{N}_{0,\infty} \mapsto \mathbb{N}$ , für die  $f_n(\infty)$  linear und  $f_n(0)$  höchstens exponentiell in  $n$  wächst, existiert eine Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$ , so daß das Spektrum der Eingabeverzögerung von  $L_n$  näherungsweise der Funktion  $f_n$  entspricht. Der Einfluß, den die Eingabeverzögerung auf die Größe von Parsern haben kann, ist somit (bis auf multiplikative Konstanten im Exponenten) vollständig geklärt. In Abschnitt 4.3 wird gezeigt, daß es eine Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$  gibt mit  $\sigma_d^Z(L_n) = n^{\Theta(1+n/(d+1))}$  für alle  $d \in \mathbb{N}_{0,\infty}$  und eine andere mit  $\sigma_d^Z(L_n) \geq n^n$  für alle  $d \in \mathbb{N}_0$  und  $\sigma_\infty^Z(L_n) = n^{\Theta(1)}$ ; in Abschnitt 4.4 werden diese Ergebnisse auch für das Spektrum der Gesamt-Zeitverzögerung gezeigt.

Diese Arbeit endet mit Kapitel 5, in dem die in dieser Arbeit offen gebliebenen Probleme sowie eine mögliche Erweiterung der Fragestellung aufgezeigt werden.

Lemmata und Sätze werden unabhängig voneinander nummeriert, und zwar fortlaufend für die gesamte Arbeit. Korollare erhalten eine zweistellige Nummer, deren erster Teil den Satz angibt, auf dem das Korollar beruht.

# Kapitel 2

## Definitionen

### 2.1 Notationen

In diesem Abschnitt werden verschiedene Notationen eingeführt, die in der gesamten Arbeit verwendet werden. Zunächst werden einige Konventionen zur Schreibweise vereinbart.

- Großbuchstaben vom Anfang des Alphabets (z. B.  $A'$ ) bezeichnen Automaten.
- Großbuchstaben vom Ende des Alphabets (z. B.  $Z_0$ ) bezeichnen Kellersymbole eines PDA.
- Sonstige Großbuchstaben (z. B.  $Q, S$ ) bezeichnen diverse Mengen.
- Griechische Großbuchstaben (z. B.  $\Gamma', \Sigma_n$ ) bezeichnen Alphabete (geordnete endliche Mengen).
- Kalligraphische Großbuchstaben (z. B.  $\mathcal{D}, \mathcal{R}, \mathcal{U}'$ ) bezeichnen in der Regel Mengen höherer Ordnung.
- Kleinbuchstaben vom Anfang des Alphabets (z. B.  $a, b$ ) bezeichnen Eingabezeichen eines PDA.
- Kleinbuchstaben vom Ende des Alphabets (z. B.  $v, w', z$ ) bezeichnen Wörter über dem Eingabealphabet eines PDA.

- Griechische Kleinbuchstaben vom Anfang des Alphabets (z. B.  $\alpha, \gamma_0$ ) bezeichnen Wörter über dem Kelleralphabet eines PDA.
- $\varepsilon$  bezeichnet das leere Wort (aus einer beliebigen Grundmenge).
- $\Sigma_\varepsilon$  bezeichnet  $\Sigma \cup \{\varepsilon\}$  (für eine beliebige Menge  $\Sigma$ ).
- $IN_{0,\infty}$  bezeichnet  $IN \cup \{0, \infty\}$ .

**Definition 1** Sei  $S$  eine endliche Menge. Die **Kardinalität von  $S$**  wird mit  $\#S$  bezeichnet, und die **Potenzmenge von  $S$**  mit  $2^S$ . Falls  $S$  total geordnet ist, wird die **lexikographisch kleinste Permutation von  $S$** , d. h., dasjenige Wort aus  $S^{|S|}$ , welches alle Zeichen aus  $S$  in aufsteigender Reihenfolge enthält, mit  $\langle\langle S \rangle\rangle$  bezeichnet.

**Definition 2** Für ein Wort  $x$  und ein  $k \in IN_0$  werden folgende Notationen vereinbart:

- $^{[k]}x$  bezeichnet das Präfix von  $x$  der Länge  $\min\{k, |x|\}$
- $x^{[k]}$  bezeichnet das Suffix von  $x$  der Länge  $\min\{k, |x|\}$
- $^{-[k]}x$  bezeichnet das Suffix von  $x$  der Länge  $\max\{|x| - k, 0\}$
- $x^{-[k]}$  bezeichnet das Präfix von  $x$  der Länge  $\max\{|x| - k, 0\}$
- $x_{[k]}$  bezeichnet das  $k$ -te Zeichen von  $x$  (falls  $1 \leq k \leq |x|$ , sonst undefiniert)

Die **Länge von  $x$** , d. h., die Anzahl der Zeichen von  $x$ , wird mit  $|x|$  bezeichnet; die **Menge von  $x$** , d. h., die Menge derjenigen Zeichen, die in  $x$  vorkommen, wird mit  $\langle x \rangle$  bezeichnet.

**Definition 3** Seien  $i, j \in \mathbb{Z}$ . Die **Menge der ganzen Zahlen von  $i$  bis  $j$**  wird bezeichnet mit

$$[i : j] \stackrel{\text{def}}{=} \{k \in \mathbb{Z} \mid i \leq k \leq j\}$$

**Definition 4** Sei  $f : IN_0 \rightarrow IN_0$  eine Funktion. Die **Länge von  $f$**  wird definiert als

$$|f| \stackrel{\text{def}}{=} \sup\left(\left\{k \in IN \mid f(k) \neq f(k-1)\right\} \cup \{1\}\right)$$

**Definition 5** Sei  $L \neq \emptyset$  eine Sprache, und sei  $\Sigma$  das kleinste Alphabet mit  $\Sigma^* \supseteq L$ .

Die **Menge der minimalen Wörter von  $L$**  wird definiert als

$$\min(L) \stackrel{\text{def}}{=} \{x \in L \mid \nexists y \in L, z \in \Sigma^+ : x = yz\}$$

$L$  heißt **präfixfrei**, falls  $\min(L) = L$  gilt.

$x \in \Sigma^*$  heißt ein **korrektes Präfix von  $L$** , falls gilt

$$\exists y \in \Sigma^* : xy \in L;$$

andernfalls heißt  $x$  ein **inkorrektes Präfix von  $L$** .

Die Menge der minimalen inkorrekten Präfixe von  $L$  wird definiert als

$$MIP(L) \stackrel{\text{def}}{=} \{xa \mid x \in \Sigma^*, a \in \Sigma, (\exists y \in \Sigma^* : xy \in L), (\nexists y \in \Sigma^* : xay \in L)\}$$

Die Nerode-Relation von  $L$  wird definiert als

$$\equiv_L \stackrel{\text{def}}{=} \{(x, y) \in \Sigma^* \times \Sigma^* \mid \forall z \in \Sigma^* : (xz \in L \iff yz \in L)\}$$

## 2.2 Deterministische Kellerautomaten

Im folgenden werden wichtige Begriffe im Zusammenhang mit DPDAs definiert. Die meisten Definitionen entstammen der vorhandenen Literatur [11, 14].

**Definition 6** Ein deterministischer Kellerautomat (kurz: DPDA) ist ein 6-tupel

$$A = (Q, \Sigma, \Gamma, \delta, C_0, F)$$

Hierbei ist

- $Q$  eine endliche Menge von **Zuständen**,
- $\Sigma$  eine endliche Menge von **Eingabezeichen**,
- $\Gamma$  eine endliche Menge von **Kellersymbolen**,
- $C_0 \in Q \times \Gamma$  die **Startkonfiguration**,
- $F \subseteq Q$  die Menge der **akzeptierenden Zustände**,
- $\delta$  eine (partielle) Abbildung von  $(Q \times \Gamma) \times \Sigma_\epsilon$  nach  $Q \times \Gamma^*$

Damit in jeder Konfiguration der folgende Übergang eindeutig festgelegt ist, gilt die Einschränkung, daß für einen Modus  $M \in Q \times \Gamma$  nicht gleichzeitig  $\delta(M, \varepsilon)$  und  $\delta(M, a)$  für ein  $a \in \Sigma$  definiert sein dürfen. Statt  $\delta((q, Z), a)$  wird meist nur  $\delta(q, Z, a)$  geschrieben.

**Definition 7** Sei  $A = (Q, \Sigma, \Gamma, \delta, C_0, F)$  ein DPDA.

Eine **Konfiguration** von  $A$  ist ein 2-tupel

$$(q, \alpha) \in Q \times \Gamma^*,$$

wobei  $q$  der Zustand und  $\alpha$  der Kellerinhalt von  $A$  ist. Das oberste Kellersymbol wird hierbei ganz rechts hingeschrieben.

Die **Länge von**  $(q, \alpha)$  wird definiert als

$$|(q, \alpha)| \stackrel{\text{def}}{=} |\alpha|$$

Der  **$k$ -Modus von**  $(q, \alpha)$  wird definiert als

$$(q, \alpha)^{[k]} \stackrel{\text{def}}{=} (q, \alpha^{[k]})$$

Der 1-Modus einer Konfiguration wird oftmals nur als ihr Modus bezeichnet.

Die **Übergangsrelation**  $\vdash_A \in (Q \times \Gamma^+) \times \Sigma_\varepsilon \times (Q \times \Gamma^*)$  ist eine dreistellige Relation, die wie folgt definiert ist: Für alle  $p, q \in Q$ ,  $Z \in \Gamma$ ,  $\alpha, \gamma \in \Gamma^*$  und  $a \in \Sigma_\varepsilon$  gelte

$$(q, \alpha Z) \vdash_A^a (p, \alpha \gamma) \stackrel{\text{def}}{\iff} \delta(q, Z, a) = (p, \gamma)$$

$\vdash_A^+$  bezeichnet die transitive Hülle und  $\vdash_A^*$  die reflexiv-transitive Hülle von  $\vdash_A$ , wobei die Eingaben der einzelnen Übergänge konkateniert werden. Falls keine Verwechslung möglich ist, kann die Angabe von  $A$  unterbleiben.

Ein Übergang  $C \vdash_A^a C'$  heißt

- **Lese-Übergang**, falls  $a \neq \varepsilon$  ist.

- $\varepsilon$ -Übergang, falls  $a = \varepsilon$  ist.

Eine Folge von Übergängen wird als **Berechnung** bezeichnet.

Eine Berechnung  $C \xrightarrow{w}^* C'$  heißt

- **originäre Berechnung**, falls  $C = C_0$  ist.
- **Scan**, geschrieben  $C \downarrow \xrightarrow{w}^* C'$ , falls  $|C| = |C'|$  ist, und falls jede Konfiguration in der Berechnung mindestens so lang ist wie  $C$ .

Ein 1-Modus  $M \in Q \times \Gamma_\varepsilon$  heißt

- **Stop-Modus**, falls  $|M| = 0$  ist, oder falls  $\delta(M, a)$  für kein  $a \in \Sigma_\varepsilon$  definiert ist.
- **Lese-Modus**, falls  $\delta(M, a)$  für ein  $a \in \Sigma$  definiert ist.

Ein Lese-Modus  $M$  heißt **vollständig**, falls  $\delta(M, a)$  für alle  $a \in \Sigma$  definiert ist; andernfalls heißt er **unvollständig**.

- $\varepsilon$ -Modus, falls  $\delta(M, \varepsilon)$  definiert ist.
- **Looping-Modus**, falls  $M \xrightarrow{\varepsilon}^+ C'$  gilt für eine Konfiguration  $C'$ , deren Modus gleich  $M$  ist.
- **fortsetzbar**, falls  $M$  ein Lese-Modus oder ein  $\varepsilon$ -Modus ist.

Diese Definitionen sind sinngemäß auf  $k$ -Modi und Konfigurationen übertragbar.

Die Menge der fortsetzbaren  $k$ -Modi von  $A$  wird mit  $(A)^{[k]}$  bezeichnet.

Jeder Modus ist entweder ein Stop-Modus oder ein Lese-Modus oder ein  $\varepsilon$ -Modus, wobei die Looping-Modi eine Teilmenge der  $\varepsilon$ -Modi sind.

**Definition 8** Sei  $A = (Q, \Sigma, \Gamma, \delta, C_0, F)$  ein DPDA.

Im folgenden werden mehrere Erweiterungen der Übergangsfunktion  $\delta$  vereinbart. Für alle  $C \in Q \times \Gamma^+$ ,  $w \in \Sigma^*$  und  $L \subseteq \Sigma^*$  definiere

$$\begin{aligned}
\delta_-(C, w) &\stackrel{\text{def}}{=} \begin{cases} C, & \text{falls } w = \varepsilon \\ C', & \text{falls es eine Berechnung } C \xrightarrow{w^+} C' \text{ gibt,} \\ & \text{deren letzter Übergang ein Lese-Übergang ist} \\ & \text{(für } w \neq \varepsilon) \end{cases} \\
\delta_+(C, w) &\stackrel{\text{def}}{=} C', \quad \text{falls } C \xrightarrow{w^*} C' \text{ gilt für eine Lese-Konf. oder Stop-Konf. } C' \\
\hat{\delta}(C, w) &\stackrel{\text{def}}{=} \{C' \in Q \times \Gamma^* \mid C \xrightarrow{w^*} C'\} \\
\hat{\delta}(C, L) &\stackrel{\text{def}}{=} \bigcup_{w \in L} \hat{\delta}(C, w)
\end{aligned}$$

Die von  $A$  durch **akzeptierende Zustände akzeptierte Sprache** wird definiert als

$$L(A) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \hat{\delta}(C_0, w) \cap (F \times \Gamma^*) \neq \emptyset\}$$

Die von  $A$  durch **leeren Keller akzeptierte Sprache** wird definiert als

$$N(A) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \hat{\delta}(C_0, w) \cap (Q \times \{\varepsilon\}) \neq \emptyset\}$$

Eine Konfiguration  $C'$  heißt

- **akzeptierend**, falls  $C' \in F \times \Gamma^*$  ist.
- **erreichbar**, falls  $C' \in \hat{\delta}(C_0, \Sigma^*)$  ist.
- **ableitbar**, falls  $\hat{\delta}(C', \Sigma^*)$  eine akzeptierende Konfiguration enthält.
- **Aufwärtskonfiguration der Berechnung**  $C \xrightarrow{w}^* C' \xrightarrow{w'}^* C''$ , falls alle Konfigurationen nach  $C'$  länger sind als  $C'$ . Die letzte Konfiguration  $C''$  ist nach dieser Definition immer eine Aufwärtskonfiguration.

Der DPDA  $A$  heißt

- **moderat**, falls  $|\delta(M, a)| \leq 2$  gilt für alle  $M \in Q \times \Gamma$  und  $a \in \Sigma_\varepsilon$ .
- **Realzeit-DPDA**, falls  $A$  keinen  $\varepsilon$ -Modus hat.

In Definition 9 werden zwei Größenmaße für DPDAs eingeführt; in Abschnitt 3.1 erfolgt eine Diskussion der jeweiligen Vor- und Nachteile.

**Definition 9** Sei  $A = (Q, \Sigma, \Gamma, \delta, C_0, F)$  ein DPDA.

Die **Produkt-Größe von  $A$**  wird definiert als

$$\|A\| \stackrel{\text{def}}{=} \#Q \cdot \#\Gamma \cdot \#\Sigma$$

Die **Größe von  $A$**  wird definiert als

$$|A| \stackrel{\text{def}}{=} \#Q + \#\Gamma + \#\Sigma + \sum_{\substack{M \in Q \times \Gamma \\ a \in \Sigma_\varepsilon \\ \delta(M,a) \text{ ist definiert}}} (|\delta(M,a)| + 4)$$

**Definition 10** Sei  $L$  eine Sprache.

$L$  ist eine **deterministisch kontextfreie Sprache** (kurz: **DCFL**), falls  $L = L(A)$  gilt für einen DPDA  $A$ . Die **Klasse der DCFLs** wird mit  $\mathcal{D}$  bezeichnet, die **Klasse der DPDAs** mit  $\mathcal{D}$ .

$L$  ist eine **Realzeit-DCFL**, falls  $L = L(A)$  gilt für einen Realzeit-DPDA  $A$ . Die **Klasse der Realzeit-DCFLs** wird mit  $\mathcal{R}$  bezeichnet, die **Klasse der Realzeit-DPDAs** mit  $\mathcal{R}$ .

$L$  ist eine **strikte Realzeit-DCFL**, falls  $L = N(A)$  gilt für einen Realzeit-DPDA  $A$ . Die **Klasse der strikten Realzeit-DCFLs** wird mit  $\mathcal{R}_0$  bezeichnet, die **Klasse der Realzeit-DPDAs**, die mit leerem Keller akzeptieren, mit  $\mathcal{R}_0$ .

Die **Klasse der regulären Sprachen** wird mit  $\mathcal{REG}$  bezeichnet.

**Definition 11** Für zwei Sprachklassen  $\mathcal{F}_1$  und  $\mathcal{F}_2$  mit  $\mathcal{F}_1 \supseteq \mathcal{F}_2$  wird das Problem, zu entscheiden, ob eine Sprache aus  $\mathcal{F}_1$  in  $\mathcal{F}_2$  ist, bezeichnet mit

$$\text{SUBCLASS}(\mathcal{F}_1, \mathcal{F}_2)$$

Für zwei Automatenklassen  $\mathcal{F}_1$  und  $\mathcal{F}_2$  wird das Problem, zu entscheiden, ob zwei Automaten aus  $\mathcal{F}_1$  und  $\mathcal{F}_2$  dieselbe Sprache akzeptieren, bezeichnet mit

$$\text{EQUIV}(\mathcal{F}_1, \mathcal{F}_2)$$

## 2.3 Verzögerungsspektren

Ein DPDA ist so definiert, daß er Eingaben *akzeptieren* kann; er ist jedoch nicht darauf ausgelegt, falsche Eingaben explizit zurückzuweisen. Um diesem Mangel abzuwehren, wird in diesem Abschnitt ein zurückweisender DPDA definiert, der eine Sprache  $L$  akzeptieren und inkorrekte Präfixe von  $L$  zurückweisen kann. Um korrekt arbeiten zu können, muß ein zurückweisender DPDA in der Lage sein, akzeptierende, zurückweisende und „neutrale“ Konfigurationen zu unterscheiden. Vor diesem Hintergrund erscheint die Akzeptierung ausschließlich mit akzeptierenden Zuständen oder mit leerem Keller zunächst ungeeignet,

da jeweils nur zwei Fälle unterschieden werden können. Bei Akzeptierung mit leerem Keller *und* akzeptierenden Zuständen ist es hingegen möglich, alle Fälle zu unterscheiden, indem ein Wort durch leeren Keller zusammen mit einem akzeptierenden Zustand *akzeptiert* wird, während es durch leeren Keller zusammen mit einem nichtakzeptierenden Zustand *zurückgewiesen* wird. Die Einschränkung, daß dieses Akzeptierungsverfahren an sich nur für präfixfreie Sprachen geeignet ist, kann umgangen werden, indem die Eingabe jeweils mit einem besonderen Zeichen abgeschlossen wird. Es ergibt sich aber das Problem, daß ein solcher DPDA nicht sofort mitteilen kann, wenn er eine Eingabe als inkorrekt erkannt hat, da er gezwungen ist, zunächst den gesamten Keller (mit  $\varepsilon$ -Übergängen) zu leeren. Da diese  $\varepsilon$ -Übergänge bei der Bestimmung der Zeitverzögerung des DPDA mitgezählt werden (vergleiche Definition 14 auf Seite 18), ist dieses Akzeptierungsverfahren hier nicht anwendbar.

Als einfache Lösung bietet sich an, die Definition eines DPDA so zu erweitern, daß die Zustände anstatt in zwei nun in drei disjunkte Klassen unterteilt werden. Es gibt die Menge der akzeptierenden Zustände, einen zurückweisenden Zustand sowie die Menge der „neutralen“ Zustände. Auf dem zurückweisenden Zustand sind keine Übergänge definiert, d. h., sobald der DPDA einmal in diesen Zustand übergegangen ist, stoppt er, ohne die restliche Eingabe zu lesen.

**Definition 12** *Ein zurückweisender DPDA (Rejecting DPDA, kurz: RDPDA) ist ein 7-tupel*

$$A = (Q, \Sigma, \Gamma, \delta, C_0, F, q_r),$$

wobei  $(Q, \Sigma, \Gamma, \delta, C_0, F)$  ein DPDA ist;  $q_r \in Q - F$  ist der zurückweisende Zustand, auf dem keine Übergänge definiert sind.

Die von  $A$  zurückgewiesene Sprache wird definiert als

$$\bar{L}(A) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \hat{\delta}(C_0, w) \cap (\{q_r\} \times \Gamma^*) \neq \emptyset\}$$

Eine Konfiguration  $(q, \alpha) \in Q \times \Gamma^*$  heißt **zurückweisend**, falls  $q = q_r$  ist.

Eine Berechnung  $C \xrightarrow{w} C'$  heißt **akzeptierend** bzw. **zurückweisend**, falls die Konfiguration  $C'$  akzeptierend bzw. zurückweisend ist.

Der Unterschied zwischen einem DPDA, der mit akzeptierenden Zuständen akzeptiert, und einem RDPDA besteht also darin, daß beim RDPDA ein Zustand als zurückweisender Zustand ausgezeichnet ist. Da der RDPDA ein Spezialfall des DPDA ist, sind alle Definitionen zu DPDAs auf RDPDAs übertragbar.

Ein RDPDA soll die Funktionsweise eines Parsers simulieren, der eine Eingabe entweder akzeptiert oder zurückweist. Deshalb ist es wünschenswert, daß ein RDPDA jede Eingabe so weit liest, bis er entweder an ihr Ende gelangt ist, oder bis er einen Fehler (ein inkorrektes Präfix der Sprache) entdeckt hat. Er sollte ein Wort  $w$  also nur dann *nicht* vollständig lesen, wenn er bereits ein echtes Präfix von  $w$  zurückgewiesen hat. Außerdem sollte ein RDPDA in einer akzeptierenden Konfiguration entweder stoppen (falls er bereits die gesamte Eingabe gelesen hat) oder ein weiteres Eingabezeichen lesen. Diese Überlegungen motivieren die folgende Definition.

**Definition 13** *Ein RDPDA  $A$  besitzt die*

- **Parsing-Eigenschaft**, *falls er weder einen Looping-Modus noch einen unvollständigen Lese-Modus hat, und falls alle erreichbaren Stop-Modi zurückweisend sind.*
- **Terminierungs-Eigenschaft**, *falls er keinen akzeptierenden  $\varepsilon$ -Modus hat.*

$A$  **erkennt** die Sprache  $L$ , falls  $L(A) = L$  gilt, und falls  $A$  die Parsing-Eigenschaft sowie die Terminierungs-Eigenschaft besitzt.

Wenn  $A$  eine Sprache  $L$  *erkennt*, dann weist er nur inkorrekte Präfixe von  $L$  zurück, da er aufgrund der Terminierungs-Eigenschaft nach dem Erreichen einer akzeptierenden Konfiguration nicht mit  $\varepsilon$ -Übergängen in eine zurückweisende Konfiguration übergehen kann. Es ist zu beachten, daß der Begriff des Erkennens einer Sprache hier weniger streng gehandhabt wird, als dies sonst im Zusammenhang mit Turingmaschinen geschieht.

Nun sind alle Hilfsmittel vorhanden, um die Verzögerung eines RDPDA zu definieren. Hierbei erscheint es sinnvoll, drei Verzögerungsmaße zu unterscheiden:

- Ein RDPDA  $A$ , der  $L$  mit *Eingabeverzögerung*  $d$  erkennt, darf nach dem Erreichen einer nicht ableitbaren Konfiguration höchstens  $d$  weitere Eingabezeichen lesen.

- Ein RDPDA  $A$ , der  $L$  mit *Zeitverzögerung*  $d$  erkennt, darf nach dem Erreichen einer nicht ableitbaren Konfiguration kein Eingabezeichen mehr lesen und höchstens  $d$   $\varepsilon$ -Übergänge ausführen.
- Ein RDPDA  $A$ , der  $L$  mit *Gesamt-Zeitverzögerung*  $d$  erkennt, darf auf nicht ableitbaren Konfiguration überhaupt keinen Übergang und in einer akzeptierenden Berechnung höchstens  $d$   $\varepsilon$ -Übergänge ausführen.

Definition 14 teilt die Übergänge von RDPDAs in vier Äquivalenzklassen (Typen) ein und definiert die soeben informal eingeführten Begriffe.

**Definition 14** Sei  $L$  eine Sprache, sei  $A$  ein RDPDA, der  $L$  erkennt, und sei  $d \in \mathbb{N}_{0,\infty}$ .

Ein Übergang  $C \xrightarrow{a} C'$  ist vom

**Typ 0**, falls gilt:  $a \neq \varepsilon$ ,  $C$  ist ableitbar

**Typ 1**, falls gilt:  $a = \varepsilon$ ,  $C$  ist ableitbar

**Typ 2**, falls gilt:  $a = \varepsilon$ ,  $C$  ist nicht ableitbar

**Typ 3**, falls gilt:  $a \neq \varepsilon$ ,  $C$  ist nicht ableitbar

$A$  erkennt  $L$  mit **Eingabeverzögerung**  $d$ , falls keine originäre Berechnung von  $A$  mehr als  $d$  Übergänge vom Typ 3 enthält.

$A$  erkennt  $L$  mit **Zeitverzögerung**  $d$ , falls keine originäre Berechnung von  $A$  einen Übergang vom Typ 3 oder mehr als  $d$  Übergänge vom Typ 2 enthält.

$A$  erkennt  $L$  mit **Gesamt-Zeitverzögerung**  $d$ , falls keine originäre Berechnung von  $A$  einen Übergang vom Typ 3 oder 2 oder mehr als  $d$  Übergänge vom Typ 1 enthält.

Wenn die akzeptierte Sprache nicht von Interesse ist, kann ihre Angabe unterbleiben, und man sagt lediglich,  $A$  erkenne mit Eingabeverzögerung  $d$  usw.

In Definition 14 wurden die Übergänge eines DPDA in vier Typen eingeteilt, wobei Typ 0 die „unverzichtbaren“ Übergänge beschreibt, während die Typen 1, 2 und 3 solche Übergänge beschreiben, die das Erkennen von fehlerhaften Eingaben verzögern. Um das Verzögerungsverhalten eines (Rejecting) DPDA vollständig zu beschreiben, müßte die maximale Anzahl von Übergängen der Typen 1, 2 und 3, die der Automat in einer originären

Berechnung ausführt, einzeln angegeben werden; dies würde jedoch ein dreidimensionales Verzögerungsmaß erfordern. Um praktikablere Verzögerungsmaße zu erhalten, wurden aus dem (nicht explizit definierten) dreidimensionalen Verzögerungsmaß drei für die Praxis besonders interessant erscheinende, eindimensionale Verzögerungsmaße abgeleitet. Ausgehend davon werden nun Verzögerungsspektren für deterministisch kontextfreie Sprachen definiert.

**Definition 15** Sei  $L$  eine DCFL.

Das **Spektrum der Eingabeverzögerung von  $L$**  ist eine unendliche Folge mit Endpunkt

$$\sigma^E(L) \stackrel{\text{def}}{=} \left( \sigma_0^E(L) \mid \sigma_1^E(L) \mid \cdots \mid \sigma_d^E(L) \mid \cdots \parallel \sigma_\infty^E(L) \right),$$

wobei für alle  $d \in \mathbb{N}_{0,\infty}$  gilt:

$$\sigma_d^E(L) \stackrel{\text{def}}{=} \min \left\{ |A| \mid A \text{ ist ein RDPDA, der } L \text{ mit Eingabeverz. } d \text{ erkennt} \right\}$$

Das **Spektrum der Zeitverzögerung von  $L$**  ist eine unendliche Folge mit Endpunkt

$$\sigma^Z(L) \stackrel{\text{def}}{=} \left( \sigma_0^Z(L) \mid \sigma_1^Z(L) \mid \cdots \mid \sigma_d^Z(L) \mid \cdots \parallel \sigma_\infty^Z(L) \right),$$

wobei für alle  $d \in \mathbb{N}_{0,\infty}$  gilt:

$$\sigma_d^Z(L) \stackrel{\text{def}}{=} \min \left\{ |A| \mid A \text{ ist ein RDPDA, der } L \text{ mit Zeitverz. } d \text{ erkennt} \right\}$$

Das **Spektrum der Gesamt-Zeitverzögerung von  $L$**  ist eine unendliche Folge mit Endpunkt

$$\sigma^G(L) \stackrel{\text{def}}{=} \left( \sigma_0^G(L) \mid \sigma_1^G(L) \mid \cdots \mid \sigma_d^G(L) \mid \cdots \parallel \sigma_\infty^G(L) \right),$$

wobei für alle  $d \in \mathbb{N}_{0,\infty}$  gilt:

$$\sigma_d^G(L) \stackrel{\text{def}}{=} \min \left( \{ \infty \} \cup \left\{ |A| \mid A \text{ ist ein RDPDA, der } L \text{ mit Gesamt-Zeitv. } d \text{ erkennt} \right\} \right)$$

In Abschnitt 3.2 wird gezeigt, daß es zu jeder DCFL  $L$  einen RDPDA gibt, der  $L$  mit Zeitverzögerung 0 erkennt. Dies begründet, daß lediglich bei der Definition des Spektrums der Gesamt-Zeitverzögerung unendliche Werte berücksichtigt werden.

Auf Verzögerungsspektren sind arithmetische Operationen und Vergleiche möglich. Vergleiche sind elementweise definiert; d. h., daß ein Verzögerungsspektrum genau dann kleiner ist als ein anderes, wenn dies für alle Elemente gilt. Addition sowie Multiplikation mit einem Skalar sind wie für Vektoren definiert. In Definition 16 wird eine vereinfachte Notation zur gleichzeitigen Angabe von oberen und unteren Schranken für Verzögerungsspektren vereinbart.

**Definition 16** Seien  $\theta = (\theta_0|\theta_1|\cdots|\theta_\infty)$  sowie  $\theta' = (\theta'_0|\theta'_1|\cdots|\theta'_\infty)$  zwei Verzögerungsspektren. Definiere

$$\left( \begin{array}{c|c|c} \theta_0 & \theta_1 & \theta_\infty \\ \theta'_0 & \theta'_1 & \theta'_\infty \end{array} \right) \stackrel{\text{def}}{=} \{(\sigma_0|\sigma_1|\cdots|\sigma_\infty) \mid \forall d \in \mathbb{N}_{0,\infty} : \theta_d \geq \sigma_d \geq \theta'_d\}$$

Die Definition gilt sinngemäß, falls einzelne Werte von  $\theta$  bzw.  $\theta'$  nicht angegeben sind.

# Kapitel 3

## Grundlegende Ergebnisse

In diesem Kapitel werden grundlegende Ergebnisse zu RDPDAs gezeigt. In Abschnitt 3.1 werden die beiden Größenmaße für DPDAs verglichen, in Abschnitt 3.2 werden Schranken für Verzögerungsspektren hergeleitet, und Abschnitt 3.3 beschäftigt sich mit der Berechenbarkeit von Verzögerungsspektren.

### 3.1 Größenmaße für deterministische Kellerautomaten

In Definition 9 auf Seite 14 wurden zwei Größenmaße für DPDAs definiert, deren jeweilige Vor- und Nachteile in diesem Abschnitt erörtert werden. Ein Größenmaß für DPDAs soll die Anzahl der Zeichen, die nötig sind, um den DPDA als Programm hinzuschreiben, möglichst genau angeben. Daneben ist es wünschenswert, daß das Größenmaß einfach ist. Die Größe  $|A|$ , die sich an die Definition von  $|A|$  in [11, Section 5.5] anlehnt, gibt die Länge des DPDA  $A$ , als Programm hingeschrieben, genau an; die „Exaktheit“ dieses Größenmaßes wird allerdings durch eine ziemlich komplexe Definition erkauft. Die Produkt-Größe  $\|A\|$  ist hingegen ein einfaches Größenmaß, das jedoch den Nachteil hat, die Länge von  $A$  möglicherweise deutlich zu unterschätzen, wenn  $A$  nicht moderat ist. In früheren Arbeiten zur Größe von (D)PDAs hat man sich in der Regel auf die Betrachtung von moderaten Automaten beschränkt [8, 11, 16]. Es ist leicht nachzuprüfen, daß hierfür das folgende Lemma gilt.

**Lemma 1** *Es gibt eine Konstante  $c > 0$ , so daß für jeden moderaten DPDA  $A$  gilt*

$$c \cdot |A| \leq \|A\| \leq |A|^3$$

In den meisten Fällen ist die Beschränkung auf moderate DPDAs problemlos möglich, denn jeder DPDA  $A$  läßt sich durch einen äquivalenten moderaten DPDA  $A'$  simulieren, für den  $|A'| = O(|A|)$  gilt [14, Lemma 10.1]. Die Äquivalenz besagt hierbei aber nur, daß  $A'$  dieselbe Sprache akzeptiert wie  $A$ ; so ist es im allgemeinen unvermeidlich, daß  $A'$  gegenüber  $A$  zusätzliche  $\varepsilon$ -Übergänge ausführt. Im Rahmen dieser Arbeit kann man  $A$  und  $A'$  jedoch nicht als äquivalent bezeichnen, da  $\varepsilon$ -Übergänge bei der Bestimmung der Zeitverzögerung und der Gesamt-Zeitverzögerung eines DPDA mitgezählt werden. Die Beschränkung auf moderate DPDAs würde also möglicherweise eine gravierende Einschränkung darstellen, und deshalb sind weiterhin *beliebige* DPDAs zugelassen.

Nun stellt sich die Frage, ob es einen wesentlichen Unterschied macht, welches der beiden Größenmaße man bei der Definition der Verzögerungsspektren zugrundelegt. Satz 1 bejaht diese Frage: Es gibt eine Sprachfamilie  $(L'_n)_{n \in \mathbb{N}}$ , so daß die Größe der minimalen Correct-Prefix-Parser für  $L'_n$  exponentiell wächst, ihre Produkt-Größe aber nur polynomiell.

Alle weiteren Ergebnisse in dieser Arbeit gelten jedoch nicht nur dann, wenn die *Größe von beliebigen RDPDAs* zur Definition der Verzögerungsspektren herangezogen wird, sondern auch für die *Produkt-Größe von moderaten RDPDAs*. Dies erleichtert insbesondere die Vergleichbarkeit mit Arbeiten wie [8, 16]. Die Übertragbarkeit der *unteren* Schranken ist offensichtlich: Wenn jeder RDPDA, der bestimmte Bedingungen erfüllt, eine gewisse Größe nicht unterschreiten kann, dann gilt das erst recht für alle moderaten RDPDAs, die diese Bedingungen erfüllen. Nach Lemma 1 ist die Produkt-Größe eines moderaten RDPDA aber — bis auf eine multiplikative Konstante — nicht kleiner als seine Größe. Die Übertragbarkeit der *oberen* Schranken auf die Produkt-Größe von moderaten DPDAs ist deshalb möglich, da die in Abschnitt 3.2 angegebenen Konstruktionen die „Moderatheit“ erhalten, und da alle in Kapitel 4 konstruierten RDPDAs bereits moderat sind.

In Satz 1 stellt sich erstmals das Problem, die Korrektheit eines (Rejecting) DPDA zu zeigen. Streng formale Beweise sind aufgrund der Definition eines DPDA als 6-tupel, dessen essentieller Bestandteil eine Übergangsfunktion von einem umständlich erscheinenden Typ ist, die noch nicht einmal zur direkten Beschreibung der akzeptierten Sprache genutzt werden kann, in der Regel sehr kompliziert und schwer verständlich<sup>1</sup>. In der vorliegenden Arbeit wird folgendermaßen vorgegangen, um die Korrektheit eines RDPDA zu zeigen: In der Regel wird zunächst die zugrundeliegende Idee kurz erläutert, dann wird der Automat formal definiert, und zuletzt folgt eine detaillierte Erklärung der Funktionsweise, die insbesondere angibt, wann der RDPDA sich in welchem Zustand befindet, und welche Informationen wann auf den Keller geschrieben werden.

**Satz 1** *Es gibt eine Familie von (regulären, präfixfreien) Sprachen  $(L'_n)_{n \in \mathbb{N}}$  und eine Konstante  $c > 0$ , so daß für alle  $n \geq 2$  gilt:*

1. *Es gibt einen moderaten RDPDA  $A$ , der  $L'_n$  mit Eingabeverzögerung  $\infty$  erkennt, mit*

$$|A| \leq n^c$$

2. *Es gibt einen RDPDA  $A$ , der  $L'_n$  mit Eingabeverzögerung 0 erkennt, mit*

$$\|A\| \leq n^c$$

3. *Für jeden RDPDA  $A$ , der  $L'_n$  mit endlicher Eingabeverzögerung erkennt, gilt*

$$|A| \geq n^n$$

**Beweis** Definiere für alle  $n \in \mathbb{N}$

$$\Sigma_n \stackrel{\text{def}}{=} [1 : n]$$

$$\Sigma'_n \stackrel{\text{def}}{=} [1 : n] \times [1 : n]$$

$$L_n \stackrel{\text{def}}{=} \left\{ a_1 \cdots a_n [i, b] v [j, c] \mid a_1, \dots, a_n \in \Sigma_n, v \in \Sigma_n^*, [i, b], [j, c] \in \Sigma'_n, \right. \\ \left. a_i = b, a_j = c \right\}$$

$$L'_n \stackrel{\text{def}}{=} L_{4n+12}$$

---

<sup>1</sup>In [9] wird aus diesem Grund eine Notation für PDAs vorgeschlagen, in der der reguläre Teil des Automaten und die Kellerbewegungen getrennt beschrieben werden.

Die Aussagen (1) und (3) folgen direkt aus Korollar 7.1 auf Seite 58, da die obige Definition von  $L'_n$  mit derjenigen dort übereinstimmt.

Zunächst wird die Idee für den Beweis von Aussage (2) erläutert, wobei die Unterscheidung zwischen  $L_n$  und  $L'_n$  eine einfachere Formulierung des Beweises ermöglicht. Es wird ein RDPDA  $A$  konstruiert, der  $L_n$  mit Eingabeverzögerung 0 erkennt. Bei der Konstruktion von  $A$  ist  $T_n$ , ein vollständiger externer Suchbaum vom Grad  $n$  der Tiefe  $n$ , entscheidend. Jeder innere Knoten von  $T_n$  hat genau  $n$  Nachfolger, und alle  $n^n$  Blätter sind auf Ebene  $n$  (die Ebene der Wurzel wird als Ebene 0 definiert).  $T_n$  enthält für jedes Wort  $w \in \Sigma_n^{\leq n}$  genau einen Knoten der Tiefe  $|w|$ . Jeder Knoten von  $T_n$  außer der Wurzel besitzt einen Schlüssel, der die Suche von Unterbäumen ermöglicht, und am Blatt für ein Wort  $w \in \Sigma_n^n$  ist außerdem  $w$  zweimal gespeichert. Der Preorder-Pfad von  $T_n$ , in einer geeigneten Codierung rückwärts hingeschrieben, wird als  $t_n$  bezeichnet.

Während der Verarbeitung einer Eingabe  $a_1 \cdots a_n [i, b] v [j, c]$  schreibt  $A$  zunächst mit einem  $\varepsilon$ -Übergang das Wort  $t_n$  auf den Keller und folgt dann dem Preorder-Pfad von  $T_n$ , indem er so lange Teile von  $t_n$  vom Keller löscht, bis er zu demjenigen Blatt von  $T_n$  gelangt, welches  $a_1 \cdots a_n$  entspricht. An diesem Blatt ist  $a_1 \cdots a_n$  zweimal gespeichert, so daß  $A$  während des Lesens von  $[i, b] v [j, c]$  leicht testen kann, ob die gesamte Eingabe in  $L_n$  ist, oder nicht.

Bevor  $A$  formal definiert wird, ist es nötig,  $t_n$  genauer zu spezifizieren. Dies geschieht in untenstehender Tabelle anhand eines Beispiels für den Fall  $n = 3$ , wobei die Leerzeichen und Zeilensprünge selbstverständlich nicht in  $t_3$  enthalten sind. Der zugehörige externe Suchbaum  $T_3$  läßt sich erkennen, indem man die Tabelle von rechts nach links liest: Die Wurzel des Baumes wurde weggelassen; die ganz rechte Spalte enthält die inneren Knoten der Tiefe 1, die nächste Spalte die inneren Knoten der Tiefe 2, und daran anschließend stehen die Blätter von  $T_3$ . Der Eintrag für jeden Knoten besteht aus einem 2-tupel, dessen erste Komponente die Tiefe des Knotens angibt, während die zweite Komponente den Schlüssel des Knotens enthält<sup>2</sup>. Jedes Blatt enthält zudem das zugehörige Wort zweimal. Eine Verallgemeinerung auf beliebige  $n \in \mathbb{N}$  ist offensichtlich. Damit  $A$  den Preorder-Pfad von  $T_n$  in der richtigen Reihenfolge abarbeitet, obwohl das oberste Kellersymbol ganz rechts steht, mußte dieser Pfad von rechts nach links und die zugehörige Tabelle von rechts unten nach links oben hingeschrieben werden.

---

<sup>2</sup>Für ein 2-tupel  $[i, b]$  definiere  $\text{first}([i, b]) \stackrel{\text{def}}{=} i$  sowie  $\text{second}([i, b]) \stackrel{\text{def}}{=} b$ .

Beispiel:  $t_3$

\$	[3, 3]	[2, 3]	[1, 3]	£	[3, 3]	[2, 3]	[1, 3]	\$	[3, 3]
\$	[3, 2]	[2, 3]	[1, 3]	£	[3, 2]	[2, 3]	[1, 3]	\$	[3, 2]
\$	[3, 1]	[2, 3]	[1, 3]	£	[3, 1]	[2, 3]	[1, 3]	\$	[3, 1] [2, 3]
\$	[3, 3]	[2, 2]	[1, 3]	£	[3, 3]	[2, 2]	[1, 3]	\$	[3, 3]
\$	[3, 2]	[2, 2]	[1, 3]	£	[3, 2]	[2, 2]	[1, 3]	\$	[3, 2]
\$	[3, 1]	[2, 2]	[1, 3]	£	[3, 1]	[2, 2]	[1, 3]	\$	[3, 1] [2, 2]
\$	[3, 3]	[2, 1]	[1, 3]	£	[3, 3]	[2, 1]	[1, 3]	\$	[3, 3]
\$	[3, 2]	[2, 1]	[1, 3]	£	[3, 2]	[2, 1]	[1, 3]	\$	[3, 2]
\$	[3, 1]	[2, 1]	[1, 3]	£	[3, 1]	[2, 1]	[1, 3]	\$	[3, 1] [2, 1] [1, 3]
\$	[3, 3]	[2, 3]	[1, 2]	£	[3, 3]	[2, 3]	[1, 2]	\$	[3, 3]
\$	[3, 2]	[2, 3]	[1, 2]	£	[3, 2]	[2, 3]	[1, 2]	\$	[3, 2]
\$	[3, 1]	[2, 3]	[1, 2]	£	[3, 1]	[2, 3]	[1, 2]	\$	[3, 1] [2, 3]
\$	[3, 3]	[2, 2]	[1, 2]	£	[3, 3]	[2, 2]	[1, 2]	\$	[3, 3]
\$	[3, 2]	[2, 2]	[1, 2]	£	[3, 2]	[2, 2]	[1, 2]	\$	[3, 2]
\$	[3, 1]	[2, 2]	[1, 2]	£	[3, 1]	[2, 2]	[1, 2]	\$	[3, 1] [2, 2]
\$	[3, 3]	[2, 1]	[1, 2]	£	[3, 3]	[2, 1]	[1, 2]	\$	[3, 3]
\$	[3, 2]	[2, 1]	[1, 2]	£	[3, 2]	[2, 1]	[1, 2]	\$	[3, 2]
\$	[3, 1]	[2, 1]	[1, 2]	£	[3, 1]	[2, 1]	[1, 2]	\$	[3, 1] [2, 1] [1, 2]
\$	[3, 3]	[2, 3]	[1, 1]	£	[3, 3]	[2, 3]	[1, 1]	\$	[3, 3]
\$	[3, 2]	[2, 3]	[1, 1]	£	[3, 2]	[2, 3]	[1, 1]	\$	[3, 2]
\$	[3, 1]	[2, 3]	[1, 1]	£	[3, 1]	[2, 3]	[1, 1]	\$	[3, 1] [2, 3]
\$	[3, 3]	[2, 2]	[1, 1]	£	[3, 3]	[2, 2]	[1, 1]	\$	[3, 3]
\$	[3, 2]	[2, 2]	[1, 1]	£	[3, 2]	[2, 2]	[1, 1]	\$	[3, 2]
\$	[3, 1]	[2, 2]	[1, 1]	£	[3, 1]	[2, 2]	[1, 1]	\$	[3, 1] [2, 2]
\$	[3, 3]	[2, 1]	[1, 1]	£	[3, 3]	[2, 1]	[1, 1]	\$	[3, 3]
\$	[3, 2]	[2, 1]	[1, 1]	£	[3, 2]	[2, 1]	[1, 1]	\$	[3, 2]
\$	[3, 1]	[2, 1]	[1, 1]	£	[3, 1]	[2, 1]	[1, 1]	\$	[3, 1] [2, 1] [1, 1]

Definiere

$$A \stackrel{\text{def}}{=} (Q, \Sigma_n \cup \Sigma'_n, \Sigma'_n \cup \{\$, \mathcal{L}\}, \delta, (q_0, \$), \{q_f\}, q_r),$$

wobei

$$Q \stackrel{\text{def}}{=} \left\{ [\rightarrow, k] \mid k \in [0 : n] \right\} \cup \left\{ [\downarrow, k, b, l] \mid k \in [1 : n], b \in \Sigma_n, l \in [0 : 1] \right\} \cup \left\{ [\Downarrow, s] \mid s \in \Sigma'_n \right\} \cup \{q_0, q_f, q_r\}$$

sowie für alle  $k \in [0 : n]$ ,  $b \in \Sigma_n$ ,  $l \in [0 : 1]$ ,  $s \in \Sigma'_n$ ,  $Z \in \Sigma'_n \cup \{\$, \mathcal{L}\}$  und  $a \in \Sigma_n \cup \Sigma'_n$

$$\begin{aligned} \delta(q_0, \$, \varepsilon) &\stackrel{\text{def}}{=} ([\rightarrow, 0], t_n) \\ \delta([\rightarrow, k], Z, a) &\stackrel{\text{def}}{=} \begin{cases} ([\downarrow, k+1, a, 1], Z), & \text{falls } k < n, a \in \Sigma_n \\ ([\downarrow, a], \varepsilon), & \text{falls } k = n, a \in \Sigma'_n \\ (q_r, \varepsilon), & \text{falls } (k < n, a \in \Sigma'_n) \vee \\ & (k = n, a \in \Sigma_n) \end{cases} \\ \delta([\downarrow, k, b, l], Z, \varepsilon) &\stackrel{\text{def}}{=} \begin{cases} ([\downarrow, k, b, 1-l], \varepsilon), & \text{falls } Z = \$ \\ ([\downarrow, k, b, l], \varepsilon), & \text{falls } (Z \neq \$, l = 0) \vee \\ & (Z \neq \$, l = 1, Z \neq [k, b]) \\ ([\rightarrow, k], \varepsilon), & \text{falls } Z \neq \$, l = 1, Z = [k, b] \end{cases} \\ \delta([\downarrow, s], Z, a) &\stackrel{\text{def}}{=} \begin{cases} ([\downarrow, s], \varepsilon), a = \varepsilon, & \text{falls } Z \in \Sigma'_n, \\ & (\text{first}(Z) \neq \text{first}(s) \vee \\ & \text{second}(Z) = \text{second}(s)) \\ (q_r, \varepsilon), a = \varepsilon, & \text{falls } Z \in \Sigma'_n, \\ & \text{first}(Z) = \text{first}(s), \\ & \text{second}(Z) \neq \text{second}(s) \\ ([\downarrow, s], Z), & \text{falls } Z = \mathcal{L}, a \in \Sigma_n \\ ([\downarrow, a], \varepsilon), & \text{falls } Z = \mathcal{L}, a \in \Sigma'_n \\ (q_f, \$), a = \varepsilon, & \text{falls } Z = \$ \end{cases} \\ \delta(q_f, \$, a) &\stackrel{\text{def}}{=} (q_r, \varepsilon) \end{aligned}$$

$A$  verarbeitet eine Eingabe  $a_1 \cdots a_n [i, b] v [j, c]$  in drei Phasen: In der ersten Phase schreibt  $A$  das Wort  $t_n$  auf den Keller, liest  $a_1 \cdots a_n$  und leert dabei den Keller so weit, daß schließlich das zu  $a_1 \cdots a_n$  gehörende Blatt von  $T_n$  zuoberst auf dem Keller ist. Hierbei befindet sich  $A$  in Zuständen der Form  $[\rightarrow, k]$  oder  $[\downarrow, k, b, l]$ . Im Zustand  $[\rightarrow, k]$  hat  $A$  bereits  $k$  Zeichen aus  $\Sigma_n$  gelesen. Falls  $k < n$  ist, dann liest  $A$  das nächste Zeichen  $a \in \Sigma_n$  und geht in den Zustand  $[\downarrow, k+1, a, 1]$  über; im Fall  $k = n$  ist  $A$  bereits an

einem Blatt von  $T_n$  angelangt, liest  $[i, b] \in \Sigma'_n$  und geht nach  $[\Downarrow, [i, b]]$  über. Im Zustand  $[\Downarrow, k, b, l]$  hat  $A$  bereits  $k$  Zeichen aus  $\Sigma_n$  gelesen, deren letztes ein  $b$  war, und sucht nun nach einem Knoten der Tiefe  $k$  mit Schlüssel  $b$ , der durch das Kellersymbol  $[k, b]$  gekennzeichnet ist, indem er mit  $\varepsilon$ -Übergängen Teile des Kellerinhalts löscht. Damit  $A$  zwischen solchen Symbolen von  $t_n$ , die einen Knoten repräsentieren, und solchen, die an einem Blatt gespeicherte Information enthalten, unterscheiden kann, sind beide Arten von Symbolen jeweils durch  $\$$  getrennt. Jedes Auftreten von  $\$$  invertiert das Flag  $l$ , welches den Inhalt des obersten Kellersymbols angibt ( $l = 0$ : Daten an einem Blatt;  $l = 1$ : Knoten mit Schlüssel).

In der zweiten Phase befindet sich  $A$  im Zustand  $[\Downarrow, [i, b]]$  und überprüft zunächst durch Leeren des Kellers bis zum Symbol  $\mathcal{L}$ , ob  $a_i = b$  gilt. Anschließend liest  $A$  die restliche Eingabe  $v[j, c]$  und geht nach  $[\Downarrow, [j, c]]$  über. In der dritten Phase befindet sich  $A$  im Zustand  $[\Downarrow, [j, c]]$  und überprüft, ob  $a_j = c$  ist. Wenn alle Vergleiche erfolgreich verlaufen, dann akzeptiert  $A$  die Eingabe, ansonsten weist er sie zurück.

Offensichtlich erkennt  $A$  die Sprache  $L_n$  mit Eingabeverzögerung 0. Die Produkt-Größe von  $A$  ergibt sich zu

$$\begin{aligned} \|A\| &= \#Q \cdot \#(\Sigma'_n \cup \{\$, \mathcal{L}\}) \cdot \#(\Sigma_n \cup \Sigma'_n) \\ &= (3n^2 + n + 4) \cdot (n^2 + 2) \cdot (n + n^2) \\ &= O(n^6) \end{aligned}$$

■

## 3.2 Schranken für Verzögerungsspektren

Die Ausgangsfrage dieser Arbeit lautet, inwieweit die Größe des minimalen Parsers für eine DCFL  $L$  davon abhängt, mit welcher Verzögerung dieser fehlerhafte Eingaben zurückweist. Daraus folgt, daß nicht die absoluten Werte eines Verzögerungsspektrums von Interesse sind, sondern vielmehr der Unterschied zwischen dem kleinsten und dem größten Wert *innerhalb* des Spektrums sowie die Verteilung des Größenunterschieds auf die dazwischenliegenden Werte. In diesem Abschnitt werden Schranken für den Größenunterschied innerhalb der drei Arten von Verzögerungsspektren angegeben; aus den Ergebnissen in

Kapitel 4 geht hervor, daß zumindest diejenigen Schranken, welche die Spektren der Eingabeverzögerung und der Zeitverzögerung betreffen, scharf sind.

Zunächst wird gezeigt, daß es zu jeder DCFL  $L$  einen RDPDA gibt, der  $L$  mit Zeitverzögerung 0 erkennt. Der entscheidende Schritt dieser Konstruktion besteht darin, einen beliebigen DPDA dahingehend zu erweitern, daß er jederzeit „weiß“, ob er sich in einer ableitbaren Konfiguration befindet, oder nicht. Eine solche Konstruktion für DPDAs, die mit leerem Keller akzeptieren, ist in [5, Lemma 1.1] angegeben. Diese Konstruktion wird in Lemma 2 an die Akzeptierung mit akzeptierenden Zuständen angepaßt, wobei Ideen aus [14, Lemma 10.4] Verwendung finden.

**Lemma 2** *Es gibt eine Konstante  $c > 0$ , so daß zu jedem DPDA  $A$  ein DPDA  $A'$  existiert, für den gilt:*

1.  $L(A') = L(A)$
2.  $A'$  simuliert jeden Übergang von  $A$  durch genau einen Übergang gleichen Typs.
3. Für jede erreichbare Konfiguration von  $A'$  geht aus dem Zustand hervor, ob die Konfiguration ableitbar ist, oder nicht.
4.  $|A'| \leq c \cdot 2^{|A|/3}$

**Beweis** Sei  $L$  eine DCFL, und sei  $A = (Q, \Sigma, \Gamma, \delta, (q_0, Z_0), F)$  ein DPDA mit  $L(A) = L$ . o. B. d. A. sei  $L \neq \emptyset$ . Es wird ein DPDA  $A'$  konstruiert, der im wesentlichen  $A$  simuliert.  $A'$  gibt jedem Symbol, das er auf den Keller schreibt, zusätzliche Informationen dazu, die es ihm ermöglichen, nicht ableitbare Konfigurationen sofort zu erkennen. Hierfür wird das Kelleralphabet erweitert, so daß ein Kellersymbol von  $A'$  aus einem Paar  $[Z, S]$  besteht, wobei  $Z \in \Gamma$  und  $S \subseteq Q$  ist. Außerdem wird jeder Zustand von  $A$  um ein Flag  $i$  erweitert, welches anzeigt, ob die betreffende Konfiguration ableitbar ist ( $i = 1$ ), oder nicht ( $i = 0$ ). Dieses Flag ist zwar nicht notwendig, es erleichtert jedoch den Beweis von Lemma 4.

Wenn sich der simulierte Automat  $A$  in der Konfiguration

$$(q, X_1 X_2 \cdots X_l) \in Q \times \Gamma^+$$

befindet, dann ist der simulierende Automat  $A'$  in der Konfiguration

$$([q, i], [X_1, T_1] [X_2, T_2] \cdots [X_l, T_l]) \in (Q \times [0 : 1]) \times (\Gamma \times 2^Q)^+,$$

wobei  $T_l$  einen Zustand  $r$  genau dann enthält, wenn  $(r, X_1X_2 \cdots X_{l-1})$  ableitbar ist für  $A$ ; für  $T_1$  bis  $T_{l-1}$  gilt entsprechendes.  $T_l$  gibt an, in welche Zustände  $A$  beim Löschen des obersten Kellersymbols  $X_l$  übergehen darf, so daß die folgende Konfiguration ableitbar ist. Daher hängt  $T_l$  nur von  $X_1 \cdots X_{l-1}$  ab. Die Bedeutung des Flags  $i$  wurde bereits erläutert.

Definiere für jedes Kellersymbol  $[Z, S] \in \Gamma \times 2^Q$

$$\mathcal{Q}([Z, S]) \stackrel{\text{def}}{=} \{r \in Q \mid \hat{\delta}((r, Z), \Sigma^*) \cap ((F \times \Gamma^*) \cup (S \times \{\varepsilon\})) \neq \emptyset\}$$

$\mathcal{Q}([Z, S])$  ist für alle  $[Z, S] \in \Gamma \times 2^Q$  effektiv berechenbar, da im wesentlichen überprüft werden muß, ob gewisse deterministisch kontextfreie Sprachen leer sind; hierfür gibt es Algorithmen [14, Theorem 6.6].

Definiere nun

$$A' \stackrel{\text{def}}{=} (Q', \Sigma, \Gamma', \delta', (q'_0, Z'_0), F'),$$

wobei

$$Q' \stackrel{\text{def}}{=} Q \times [0 : 1]$$

$$\Gamma' \stackrel{\text{def}}{=} \Gamma \times 2^Q$$

$$q'_0 \stackrel{\text{def}}{=} [q_0, 1]$$

$$Z'_0 \stackrel{\text{def}}{=} [Z_0, F]$$

$$F' \stackrel{\text{def}}{=} F \times \{1\}$$

$\delta'$  wird folgendermaßen definiert:

Falls  $\delta(q, Z, a) = (p, X_1X_2 \cdots X_l)$  für ein  $q \in Q$ ,  $Z \in \Gamma$  und  $a \in \Sigma_\varepsilon$ , dann sei für alle  $i \in [0 : 1]$  und  $S \subseteq Q$

$$\delta'([q, i], [Z, S], a) \stackrel{\text{def}}{=} ([p, i'], [X_1, T_1] [X_2, T_2] \cdots [X_l, T_l]),$$

wobei

$$\begin{aligned} T_1 &\stackrel{\text{def}}{=} S \\ T_k &\stackrel{\text{def}}{=} \mathcal{Q}([X_{k-1}, T_{k-1}]) \text{ für alle } k \in [2 : l] \\ i' &\stackrel{\text{def}}{=} \begin{cases} 1, & \text{falls } (l \geq 1, p \in \mathcal{Q}([X_l, T_l])) \vee \\ & (l = 0, p \in S) \\ 0, & \text{sonst} \end{cases} \end{aligned}$$

Durch Induktion nach der Anzahl der Übergänge läßt sich folgende Behauptung zeigen, deren Beweis analog zu [14, Lemma 10.4] sowie [5, Lemma 1.1] erfolgt.

$$\begin{aligned} (q'_0, Z'_0) &\stackrel{w}{A'}^* ([p, i], [X_1, T_1] \cdots [X_l, T_l]) \\ \iff &\begin{cases} (q_0, Z_0) \stackrel{w}{A}^* (p, X_1 \cdots X_l), \\ T_1 = F, \\ \forall k \in [2 : l] : T_k = \mathcal{Q}([X_{k-1}, T_{k-1}]), \\ (i = 1 \iff (p, X_1 \cdots X_l) \text{ ist ableitbar}) \end{cases} \end{aligned}$$

$A'$  simuliert jeden Übergang von  $A$  durch genau einen Übergang gleichen Typs. Nach der Definition von  $|A|$  gilt  $|A| \geq 4 \cdot \#Q$ , falls  $A$  keine überflüssigen Zustände hat. Für die Größe von  $A'$  folgt daraus

$$\begin{aligned} |A'| &= O(|A| \cdot 2^{\#Q}) \\ &= O(2^{|A|/12} \cdot 2^{|A|/4}) \\ &= O(2^{|A|/3}) \end{aligned}$$

■

**Lemma 3** *Es gibt eine Konstante  $c > 0$ , so daß zu jedem DPDA  $A'$ , der auf nicht ableitbaren Konfigurationen keinen Übergang ausführt, ein RDPDA  $A''$  existiert, für den gilt:*

1.  $A''$  erkennt  $L(A')$  mit Zeitverzögerung 0.
2.  $|A''| \leq c \cdot |A'|^3$

**Beweis** Es wird ein RDPDA  $A''$  konstruiert, der im wesentlichen  $A'$  simuliert, der aber zusätzlich die Parsing-Eigenschaft sowie die Terminierungs-Eigenschaft hat.  $A''$  bedarf gegenüber  $A'$  folgender Erweiterungen:  $A''$  speichert in seiner endlichen Kontrolle, d. h., mit Hilfe seiner Zustände, das oberste Kellersymbol von  $A'$ , sowie ob  $A'$  seit dem Lesen des letzten Eingabezeichens in einem akzeptierenden Zustand war. Wenn der simulierte Automat  $A'$  im Modus  $(q, Z)$  mit  $Z \neq \varepsilon$  ist, dann ist der simulierende Automat  $A''$  im Zustand  $[q, Z, i]$ , wobei  $i = 1$  gilt, falls  $A'$  seit dem Lesen des letzten Eingabezeichens in einem akzeptierenden Zustand war, und  $i = 0$  andernfalls.

Bevor  $A''$  einen Übergang  $C \xrightarrow{a}_{A'} C'$  simuliert, testet er, ob  $A'$  von  $C'$  aus noch weitere Eingabezeichen lesen kann. Falls  $C'$  eine Looping-Konfiguration oder eine Stop-Konfiguration ist, dann geht  $A''$  von  $C$  aus entweder in den akzeptierenden Zustand  $f_1$  oder in den zurückweisenden Zustand  $f_0$  über, abhängig davon, ob die bis zur Konfiguration  $C'$  gelesene Eingabe in  $L$  ist, oder nicht. Falls  $C$  eine unvollständige Lese-Konfiguration von  $A'$  ist, dann geht  $A''$  von  $C$  aus mit denjenigen Eingabezeichen, für die bei  $A'$  kein Übergang definiert ist, in den zurückweisenden Zustand  $f_0$  über.

Sei  $A' = (Q', \Sigma, \Gamma', \delta', (q'_0, Z'_0), F')$  ein DPDA. Es folgt die formale Definition von  $A''$ .

Definiere den RDPDA

$$A'' \stackrel{\text{def}}{=} (Q'', \Sigma, \Gamma'', \delta'', (q''_0, Z''_0), F'', f_0),$$

wobei

$$Q'' \stackrel{\text{def}}{=} (Q' \times \Gamma' \times [0 : 1]) \cup \{f_0, f_1\}$$

$$\Gamma'' \stackrel{\text{def}}{=} \Gamma' \cup \{Z''_0\}$$

$$q''_0 \stackrel{\text{def}}{=} \begin{cases} [q'_0, Z'_0, 0], & \text{falls } q'_0 \notin F' \\ [q'_0, Z'_0, 1], & \text{falls } q'_0 \in F' \end{cases}$$

$$F'' \stackrel{\text{def}}{=} \{[q, Z, 1] \mid (q, Z) \text{ ist kein } \varepsilon\text{-Modus}\} \cup \{f_1\}$$

$\delta''$  wird durch folgende Regeln definiert:

1. Falls  $\delta'(q, Z, a) = (p, \gamma)$  mit  $p \in Q'$  und  $\gamma \in (\Gamma')^*$  für ein  $q \in Q'$ ,  $Z \in \Gamma'$  und  $a \in \Sigma_\varepsilon$ , dann sei für alle  $i \in [0 : 1]$  und  $Z' \in \Gamma''$

$$\delta''([q, Z, i], Z', a) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} ([p, (Z' \gamma)^{[1]}, i'], (Z' \gamma)^{-[1]}), \\ i' \stackrel{\text{def}}{=} \begin{cases} 1, & \text{falls } p \in F' \vee (i = 1, a = \varepsilon) \\ 0, & \text{sonst} \end{cases} \\ \text{falls } (p, Z' \gamma) \text{ weder eine Stop-Konfiguration} \\ \text{noch eine Looping-Konfiguration ist} \\ (f_{i'}, Z''_0), \\ i' \stackrel{\text{def}}{=} \begin{cases} 1, & \text{falls } (p, Z' \gamma) \text{ ableitbar ist } \vee \\ & (i = 1, a = \varepsilon) \\ 0, & \text{sonst} \end{cases} \\ \text{falls } (p, Z' \gamma) \text{ eine Stop-Konfiguration} \\ \text{oder eine Looping-Konfiguration ist} \end{array} \right.$$

(Für alle  $p \in Q'$  wird  $(p, Z''_0)$  als Stop-Konfiguration angesehen.)

2. Falls  $\delta'(q, Z, a)$  für ein  $q \in Q'$ ,  $Z \in \Gamma'$  und  $a \in \Sigma$  nicht definiert ist, obwohl  $(q, Z)$  ein Lese-Modus von  $A'$  ist, dann sei für alle  $i \in [0 : 1]$  und  $Z' \in \Gamma''$

$$\delta''([q, Z, i], Z', a) \stackrel{\text{def}}{=} (f_0, \varepsilon)$$

3. Für alle  $a \in \Sigma$  sei

$$\delta''(f_1, Z''_0, a) \stackrel{\text{def}}{=} (f_0, \varepsilon)$$

Es ist entscheidbar, ob eine Konfiguration eine Looping-Konfiguration ist, und wenn ja, ob diese ableitbar ist [2, Theorem 2.22]. Eine Stop-Konfiguration ist genau dann ableitbar, wenn sie akzeptierend ist. Daraus folgt, daß die Definition von  $\delta''$  effektiv ist.

Mit Regel 1 kann  $A''$  einen Übergang von  $A'$  simulieren. Wenn  $A'$  dabei eine Stop-Konfiguration oder eine Looping-Konfiguration erreichen würde, dann geht  $A''$  stattdessen in den Zustand  $f_1$  oder  $f_0$  über. Regel 2 erweitert jeden unvollständigen Lese-Modus

von  $A'$  zu einem vollständigen. Regel 3 bewirkt, daß  $A''$  vom neuen akzeptierenden Zustand  $f_1$  aus mit jedem Eingabezeichen in den zurückweisenden Zustand  $f_0$  übergeht. Es ist leicht zu zeigen, daß  $A''$  nach dem Lesen eines Wortes  $w$  genau dann in einen akzeptierenden Zustand (aus  $F''$ ) übergeht, wenn  $A'$  nach dem Lesen von  $w$  irgendwann in einen akzeptierenden Zustand (aus  $F'$ ) übergeht. Demnach gilt  $L(A'') = L(A')$ .

Immer dann, wenn  $A'$  eine Stop-Konfiguration, eine Looping-Konfiguration oder eine unvollständige Lese-Konfiguration erreicht und deshalb kein Eingabezeichen mehr einliest, geht  $A''$  entweder direkt oder über  $f_1$  in den zurückweisenden Zustand  $f_0$  über. Daraus folgt, daß  $A''$  die Parsing-Eigenschaft besitzt. Da  $A''$  keinen akzeptierenden  $\varepsilon$ -Modus hat, besitzt er auch die Terminierungs-Eigenschaft.  $A''$  simuliert jeden Übergang von  $A'$  nach Regel 1 durch einen Übergang gleichen Typs; außerdem führt  $A''$  eventuell einen zusätzlichen Übergang vom Typ 0 nach Regel 2 oder 3 aus. Da  $A''$  auf nicht ableitbaren Konfigurationen keinen Übergang ausführt, erkennt er  $L(A')$  mit Zeitverzögerung 0. Für die Größe von  $A''$  gilt

$$\begin{aligned} |A''| &= O\left(\#\Gamma' \cdot (|A'| + \#(A')^{[1]} \cdot \#\Sigma)\right) \\ &= O(|A'|^3) \end{aligned}$$

■

Satz 2 und Korollar 2.1 fassen die Ergebnisse der Lemmata 2 und 3 zusammen.

**Satz 2** *Es gibt eine Konstante  $c > 0$ , so daß zu jedem DPDA  $A$  ein RDPDA  $A''$  mit  $|A''| \leq c \cdot 2^{|A|}$  existiert, der  $L(A)$  mit Zeitverzögerung 0 erkennt.*

**Beweis** Sei  $L$  eine Sprache, und sei  $A$  ein DPDA mit  $L(A) = L$ . Konstruiere gemäß Lemma 2 auf Seite 31 einen DPDA  $A'$ , und beschränke dessen Übergänge auf *ableitbare* Konfigurationen. Konstruiere, ausgehend von diesem Automaten, gemäß Lemma 3 auf Seite 33 einen RDPDA  $A''$ , der  $L$  mit Zeitverzögerung 0 erkennt. ■

**Korollar 2.1** *Es gibt eine Konstante  $c > 0$ , so daß zu jedem Realzeit-DPDA  $A$  ein RDPDA  $A''$  mit  $|A''| \leq c \cdot 2^{|A|}$  existiert, der  $L(A)$  mit Gesamt-Zeitverzögerung 0 erkennt.*

In den folgenden Korollaren wird das Ergebnis aus Satz 2 als obere bzw. untere Schranke für die Spektren der Eingabeverzögerung und der Zeitverzögerung formuliert. Alle angege-

benen Schranken sind näherungsweise scharf; für Korollar 2.2 folgt dies aus Korollar 7.1, für Korollar 2.3 folgt dies aus Korollar 7.2, und für Korollar 2.4 folgt dies aus Satz 8.

**Korollar 2.2** *Es gibt eine Konstante  $c > 0$ , so daß für jede Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$  mit  $\sigma_\infty^E(L_n) \leq n$  für alle  $n \in \mathbb{N}$  gilt*

$$\sigma^E(L_n) \leq (c \cdot 2^n |c \cdot 2^n| \cdots |c \cdot 2^n| \cdots \|n)$$

**Korollar 2.3** *Es gibt eine Konstante  $c > 0$ , so daß für jede Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$  mit  $\sigma_0^E(L_n) \geq c \cdot 2^n$  für alle  $n \in \mathbb{N}$  gilt*

$$\sigma^E(L_n) \geq (c \cdot 2^n |n| \cdots |n| \cdots \|n)$$

**Korollar 2.4** *Es gibt eine Konstante  $c > 0$ , so daß für jede Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$  mit  $\sigma_\infty^Z(L_n) \leq n$  für alle  $n \in \mathbb{N}$  gilt*

$$\sigma^Z(L_n) \leq (c \cdot 2^n |c \cdot 2^n| \cdots |c \cdot 2^n| \cdots \|n)$$

Um eine annähernd scharfe untere Schranke für das Spektrum der Zeitverzögerung herzuleiten, reicht es nicht aus, extreme Werte zu betrachten, sondern es müssen auch „lokale“ Eigenschaften des Spektrums untersucht werden, so wie dies in Satz 3 und den darauffolgenden Korollaren geschieht.

**Satz 3** *Sei  $L$  eine DCFL, und sei  $d' < d$  für  $d, d' \in \mathbb{N}_0$ . Dann gilt*

1.  $\sigma_{d'}^Z(L) \leq \left(\sigma_d^Z(L)\right)^{2^{\lceil (d+1)/(d'+1) \rceil - 1}}$
2.  $\sigma_{d'}^G(L) \leq \left(\sigma_d^G(L)\right)^{2^{\lceil (d+1)/(d'+1) \rceil - 1}}$

**Beweis** Zu jedem DPDA  $A$  kann ein DPDA  $A'$  konstruiert werden, der eine Folge aus  $t$  Übergängen von  $A$ , unter denen höchstens ein Lese-Übergang ist, durch *einen* Übergang simuliert [12, Theorem 2.1]. Die Konstruktion basiert darauf, in einem Kellersymbol von  $A'$  jeweils bis zu  $2t - 1$  Kellersymbole von  $A$  zusammenzufassen. Ausgehend von einem RDPDA  $A$ , der  $L$  mit Zeitverzögerung  $d$  (Gesamt-Zeitverzögerung  $d$ ) erkennt, wird mit  $t \stackrel{\text{def}}{=} \lceil (d+1)/(d'+1) \rceil$  ein RDPDA  $A'$  konstruiert, der  $L$  mit Zeitverzögerung  $d'$  (Gesamt-Zeitverzögerung  $d'$ ) erkennt. ■

Die beiden folgenden Korollare geben obere Schranken für die Spektren der Zeitverzögerung und der Gesamt-Zeitverzögerung an. Korollar 3.1 folgt aus den Sätzen 2 und 3; aus Satz 9 ergibt sich, daß die angegebene Schranke annähernd scharf ist. Korollar 3.2 folgt aus Satz 3.

**Korollar 3.1** *Es gibt eine Konstante  $c \geq 1$ , so daß für jede Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$  mit  $\sigma_0^Z(L_n) \geq n^{cn}$  für alle  $n \in \mathbb{N}$  gilt*

$$\sigma^Z(L_n) \geq \left( n^n \mid n^{n/2} \mid n^{n/3} \mid \dots \mid n \mid n \mid \dots \parallel n \right)$$

**Korollar 3.2** *Es gibt eine Konstante  $c \geq 1$ , so daß für jede Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$  mit  $\sigma_0^G(L_n) \geq n^{cn}$  für alle  $n \in \mathbb{N}$  gilt*

$$\sigma^G(L_n) \geq \left( n^n \mid n^{n/2} \mid n^{n/3} \mid \dots \mid n^{n/d} \mid \dots \parallel 1 \right)$$

Die beiden folgenden Korollare machen genauere Aussagen über solche Spektren der Zeitverzögerung und der Gesamt-Zeitverzögerung, die zu Beginn einen exponentiellen Größenunterschied enthalten, der sich auf einen — gemäß Korollar 3.1 bzw. 3.2 — minimalen Abschnitt verteilt. Korollar 3.3 folgt aus Satz 3 und Korollar 3.1; Satz 9 zeigt, daß eine Sprachfamilie mit dem angegebenen Spektrum der Zeitverzögerung auch existiert. Korollar 3.4 folgt aus Satz 3 und Korollar 3.2; Satz 10 zeigt, daß es eine Sprachfamilie gibt, deren Spektrum der Gesamt-Zeitverzögerung im angegebenen Rahmen liegt.

**Korollar 3.3** *Es gibt Konstanten  $c, c', c'' > 0$ , so daß für jede Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$  mit  $\sigma_0^Z(L_n) \geq n^{c'n}$  und  $\sigma_n^Z(L_n) \leq n^{c''}$  für alle  $n \in \mathbb{N}$  gilt*

$$\sigma^Z(L_n) \in \left( \begin{array}{c} n^{cn} \mid n^{cn/2} \mid n^{cn/3} \mid \dots \mid n^c \mid n^c \mid \dots \parallel n^c \\ n^n \mid n^{n/2} \mid n^{n/3} \mid \dots \mid n \mid n \mid \dots \parallel n \end{array} \right)$$

**Korollar 3.4** *Es gibt Konstanten  $c, c', c'' > 0$ , so daß für jede Sprachfamilie  $(L_n)_{n \in \mathbb{N}}$  mit  $\sigma_0^G(L_n) \geq n^{c'n}$  und  $\sigma_n^G(L_n) \leq n^{c''}$  für alle  $n \in \mathbb{N}$  gilt*

$$\sigma^G(L_n) \in \left( \begin{array}{c} n^{cn} \mid n^{cn/2} \mid n^{cn/3} \mid \dots \mid n^c \mid n^c \mid \dots \parallel n^c \\ n^n \mid n^{n/2} \mid n^{n/3} \mid \dots \mid n \mid n^{n/(n+1)} \mid \dots \parallel 1 \end{array} \right)$$

Satz 4 beantwortet die Frage, welche Sprachen von einem RDPDA mit endlicher Gesamt-Zeitverzögerung erkannt werden können.

**Satz 4** Sei  $L$  eine DCFL. Dann sind folgende Aussagen äquivalent:

1.  $L$  ist eine Realzeit-DCFL.
2.  $L$  wird von einem RDPDA mit Gesamt-Zeitverzögerung 0 erkannt.
3.  $L$  wird von einem RDPDA mit endlicher Gesamt-Zeitverzögerung erkannt.

**Beweis** Der Schritt (1)  $\rightarrow$  (2) folgt aus Korollar 2.1, die Schritte (2)  $\rightarrow$  (1) sowie (2)  $\rightarrow$  (3) sind trivial, und der Schritt (3)  $\rightarrow$  (2) folgt aus Satz 3. ■

Es ist bekannt, daß  $\mathcal{R} \subset \mathcal{D}$  gilt; d. h., es gibt deterministisch kontextfreie Sprachen, die nicht von einem Realzeit-DPDA akzeptiert werden [21, Theorem 15]. Zusammen mit Satz 4 ergibt sich daraus das folgende Korollar.

**Korollar 4.1** Es gibt eine DCFL, die nicht mit endlicher Gesamt-Zeitverzögerung erkannt werden kann.

Nachdem Satz 4 diejenigen Sprachen charakterisiert hat, für die alle Werte des Spektrums der Gesamt-Zeitverzögerung endlich sind, stellt sich die Frage, ob für diese Spektren vergleichbare obere Schranken hergeleitet werden können wie für jene der Eingabeverzögerung und der Zeitverzögerung in den Korollaren 2.2 und 2.4. Im nächsten Abschnitt wird gezeigt, daß die Existenz einer rekursiven oberen Schranke für den Größenunterschied in Spektren der Gesamt-Zeitverzögerung von Realzeit-DCFLs ein Entscheidungsverfahren für das bislang ungelöste Teilklassenproblem  $SUBCLASS(\mathcal{D}, \mathcal{R})$  liefern würde, so daß eine einfache Antwort auf diese Frage nicht zu erwarten ist.

### 3.3 Berechenbarkeit von Verzögerungsspektren

In diesem Abschnitt wird das Problem der effektiven Berechenbarkeit von Verzögerungsspektren behandelt. Zunächst soll begründet werden, warum die Tatsache, daß es sich bei Verzögerungsspektren um *unendliche* Zahlenfolgen handelt, nicht per se gegen ihre effektive Berechenbarkeit spricht. Hierzu wird in [10], wo Spektren (für reguläre Sprachen) erstmals definiert werden, argumentiert, daß ein monoton fallendes, diskretes Spektrum

nur endlich viele *verschiedene* Werte enthält und durch diese Werte zusammen mit der Angabe der Grenzen zwischen je zwei verschiedenen Werten eindeutig definiert ist.

Zur Berechnung eines Verzögerungsspektrums für eine Sprache  $L$  müssen also die Größen von endlich vielen minimalen RDPDAs, die  $L$  mit einer bestimmten Verzögerung erkennen, ermittelt werden. Allem Anschein nach ist das nur dann allgemein möglich, wenn  $EQUIV(D, D)$  eine Lösung hat. Dieses Problem gehört jedoch zu den klassischen, bislang ungelösten Problemen der Automatentheorie. Trotzdem kann eine Reihe interessanter Fakten gezeigt werden.

**Lemma 4** *Für jeden RDPDA  $A$  ist es entscheidbar, ob ein  $d \in \mathbb{N}_{0,\infty}$  existiert, so daß  $A$  mit Eingabeverzögerung  $d$  (Zeitverzögerung  $d$ , Gesamt-Zeitverzögerung  $d$ ) erkennt, und wenn ja, welches das minimale solche  $d$  ist.*

**Beweis** Der Beweis verläuft folgendermaßen: Zunächst wird überprüft, ob  $A$  die Parsing-Eigenschaft sowie die Terminierungs-Eigenschaft besitzt. Falls dies zutrifft, wird ein nicht-deterministischer PDA  $A''$  konstruiert, der das Wort  $i^d$  mit  $i \in \{0, 1, 2, 3\}$  und  $d \in \mathbb{N}_{0,\infty}$  genau dann akzeptiert, wenn es eine originäre Berechnung von  $A$  gibt, die  $d$  Übergänge vom Typ 0 enthält.

Sei  $A = (Q, \Sigma, \Gamma, \delta, (q_0, Z_0), F, q_r)$  ein RDPDA. Zur Überprüfung der Parsing-Eigenschaft sind zunächst die erreichbaren Modi von  $A$  zu ermitteln, um zu testen, ob ein nicht zurückweisender Stop-Modus darunter ist. Die Frage nach der Erreichbarkeit eines Modus reduziert sich im wesentlichen auf die Frage, ob eine bestimmte DCFL leer ist und ist somit algorithmisch entscheidbar. Auch für den Test, ob  $A$  einen Looping-Modus hat, gibt es Algorithmen [2, Theorem 2.22]. Unvollständige Lese-Modi von  $A$  lassen sich unmittelbar durch Betrachtung der Übergangsfunktion  $\delta$  erkennen. Dasselbe gilt für akzeptierende  $\varepsilon$ -Modi, so daß auch die Terminierungs-Eigenschaft entscheidbar ist.

Nun ist noch der PDA  $A''$  zu konstruieren. Die zugrundeliegende Idee hierbei ist die, verschiedene Eingabezeichen als Zähler für die verschiedenen Typen von Übergängen einzusetzen. Um für jeden Übergangstyp einen Zähler zu haben, wird  $A''$  über dem Eingabealphabet  $\Sigma'' \stackrel{\text{def}}{=} \{0, 1, 2, 3\}$  definiert.

Konstruiere zunächst gemäß Lemma 2 auf Seite 31 einen DPDA

$$A' = (Q', \Sigma, \Gamma', \delta', (q'_0, Z'_0), F')$$

mit  $Q' = Q \times [0 : 1]$ , der nicht ableitbare Konfiguration sofort erkennt. Der zu konstruierende PDA  $A''$  hat denselben Kern wie  $A'$ . Damit ist gemeint, daß  $A''$  genau dann mit einem Übergang von einer Konfiguration  $C$  in eine Konfiguration  $C'$  übergehen kann, wenn dies auch für  $A'$  zutrifft. Die beiden Automaten unterscheiden sich dadurch, daß sie in den betreffenden Übergängen in der Regel andere Eingabezeichen lesen, und daß alle Zustände von  $A''$  akzeptierend sind. Es folgt die formale Definition von  $A''$ .

Definiere

$$A'' \stackrel{\text{def}}{=} (Q', \Sigma'', \Gamma', \delta'', (q'_0, Z'_0), Q')$$

$\delta''$  wird durch folgende Regeln definiert<sup>3</sup>:

**Typ 0:** Falls  $\delta'([q, 1], Z, a) = C$  für ein  $q \in Q$ ,  $Z \in \Gamma'$  und  $a \in \Sigma$ , dann sei

$$\begin{aligned} C &\in \delta''([q, 1], Z, \varepsilon) \\ C &\in \delta''([q, 1], Z, 0) \end{aligned}$$

**Typ 1:** Falls  $\delta'([q, 1], Z, \varepsilon) = C$  für ein  $q \in Q$  und  $Z \in \Gamma'$ , dann sei

$$\begin{aligned} C &\in \delta''([q, 1], Z, \varepsilon) \\ C &\in \delta''([q, 1], Z, 1) \end{aligned}$$

**Typ 2:** Falls  $\delta'([q, 0], Z, \varepsilon) = C$  für ein  $q \in Q$  und  $Z \in \Gamma'$ , dann sei

$$\begin{aligned} C &\in \delta''([q, 0], Z, \varepsilon) \\ C &\in \delta''([q, 0], Z, 2) \end{aligned}$$

**Typ 3:** Falls  $\delta'([q, 0], Z, a) = C$  für ein  $q \in Q$ ,  $Z \in \Gamma'$  und  $a \in \Sigma$ , dann sei

$$\begin{aligned} C &\in \delta''([q, 0], Z, \varepsilon) \\ C &\in \delta''([q, 0], Z, 3) \end{aligned}$$

Offensichtlich liest  $A''$  das Wort  $i^d$  genau dann vollständig, wenn es eine originäre Berechnung von  $A$  (bzw.  $A'$ ) gibt, die  $d$  Übergänge vom Typ  $i$  enthält. Damit kann die Aussage des Lemmas gezeigt werden: Falls  $A$  sowohl die Terminierungs-Eigenschaft als auch die Parsing-Eigenschaft besitzt, dann wird der PDA  $A''$  konstruiert und überprüft, ob  $L(A'') \cap 3^*$  endlich ist [14, Theorem 6.6]. Das trifft genau dann zu, wenn  $A$  mit endlicher Eingabeverzögerung erkennt. In diesem Fall wird mit dem CYK-Algorithmus [14] der

---

<sup>3</sup>Man beachte, daß  $\delta''$  keine Funktion, sondern eine allgemeine Relation ist.

Reihe nach überprüft, ob 3, 33, 333, ... von  $A''$  gelesen (und akzeptiert) werden. Wenn  $3^{d+1}$  das erste Wort ist, welches  $A''$  nicht akzeptiert, dann gibt  $d$  die (minimale) Eingabeverzögerung an, mit der  $A$  erkennt. Die Zeitverzögerung und die Gesamt-Zeitverzögerung werden entsprechend ermittelt. ■

Zur effektiven Berechnung von Werten aus einem Verzögerungsspektrum für  $L$  fehlt noch ein Algorithmus, der für zwei beliebige DPDAs entscheidet, ob sie dieselbe Sprache akzeptieren. Wie schon erwähnt wurde, ist  $EQUIV(D, D)$  jedoch noch nicht gelöst. Eines der umfassendsten Ergebnisse zu dieser Fragestellung besteht in der Lösung von  $EQUIV(D, R)$ , d. h., die Äquivalenz ist entscheidbar für zwei DPDAs, von denen einer ein Realzeit-DPDA ist [19]. In den Sätzen 5 und 6 sowie in Korollar 6.1 werden die Erkenntnisse zur Berechenbarkeit von Verzögerungsspektren zusammengefaßt.

**Satz 5**  $SUBCLASS(\mathcal{D}, \mathcal{R})$  hat eine Lösung, falls eine der beiden folgenden Aussagen wahr ist:

1. Für jede DCFL  $L$  ist  $\sigma^G(L)$  berechenbar.
2. Es gibt eine rekursive Funktion  $\varphi : \mathbb{N} \mapsto \mathbb{N}$ , so daß für jede Realzeit-DCFL  $L$  gilt

$$\sigma_0^G(L) \leq \varphi(\sigma_\infty^G(L))$$

**Beweis** Angenommen, Aussage 1 sei wahr. Dann ist für jede DCFL  $L$  entscheidbar, ob  $\sigma_0^G(L)$  endlich ist. Nach Satz 4 ist dies genau dann der Fall, wenn  $L$  eine Realzeit-DCFL ist. Daraus folgt die Entscheidbarkeit von  $SUBCLASS(\mathcal{D}, \mathcal{R})$ .

Angenommen, Aussage 2 sei wahr, und es gäbe eine Funktion  $\varphi$  wie oben angegeben. Sei  $A$  ein DPDA. Konstruiere gemäß Satz 2 einen RDPDA  $A''$ , der  $L(A)$  mit Zeitverzögerung 0 / Gesamt-Zeitverzögerung  $\infty$  erkennt. Genau dann, wenn  $L(A)$  eine Realzeit-DCFL ist, gibt es einen Realzeit-RDPDA  $A_R$  der Größe  $|A_R| \leq \varphi(|A''|)$ , der  $L(A)$  mit Gesamt-Zeitverzögerung 0 erkennt. Die Anzahl der Realzeit-RDPDA bis zur Größe  $\varphi(|A''|)$  ist endlich, und für jeden Realzeit-RDPDA kann nach Lemma 4 entschieden werden, ob er mit Gesamt-Zeitverzögerung 0 erkennt, und ob er  $L(A)$  akzeptiert. Daraus folgt die Entscheidbarkeit von  $SUBCLASS(\mathcal{D}, \mathcal{R})$ . ■

Für das Teilklassenproblem  $SUBCLASS(\mathcal{D}, \mathcal{R})$  ist noch keine Lösung bekannt; die umfassendsten Ergebnisse hierzu sind Lösungen für  $SUBCLASS(\mathcal{D}, \mathcal{REG})$  [22, 23] sowie für

*SUBCLASS*( $\mathcal{D}, \mathcal{R}_0$ ) [20]. Daraus folgt, daß auf die Frage nach der Existenz einer rekursiven oberen Schranke für den Größenunterschied in Spektren der Gesamt-Zeitverzögerung von Realzeit-DCFLs keine einfache Antwort zu erwarten ist.

**Satz 6** *Für jede DCFL  $L$  sind  $\sigma^E(L)$  und  $\sigma^Z(L)$  berechenbar unter der Annahme, daß ein Orakel existiert, welches  $EQUIV(D, D)$  entscheidet.*

**Beweis** o. B. d. A. sei  $L$  durch einen DPDA  $A$  mit  $L(A) = L$  gegeben. (Falls  $L$  durch eine kontextfreie Grammatik gegeben ist, kann aus dieser effektiv ein DPDA für  $L$  konstruiert werden.) Alle RDPDAs mit festem Eingabealphabet können in einer kanonischen Reihenfolge nach aufsteigender Größe aufgezählt werden. Für jeden solchen RDPDA  $A'$  wird zunächst das Orakel befragt, ob  $L(A') = L(A)$  ist. Falls dies zutrifft, dann wird gemäß Lemma 4 getestet, ob ein  $d \in \mathbb{N}_{0,\infty}$  existiert, so daß  $A'$  die Sprache  $L$  mit Eingabeverzögerung  $d$  (Zeitverzögerung  $d$ ) erkennt, und gegebenenfalls wird das minimale solche  $d$  berechnet. Damit sind  $\sigma_d^E(L)$  ( $\sigma_d^Z(L)$ ) und alle darauffolgenden Positionen des Spektrums festgelegt. Dieser Vorgang wird so lange wiederholt, bis ein RDPDA  $A'$  gefunden wird, der  $L$  mit Eingabeverzögerung 0 (Zeitverzögerung 0) erkennt und die offen gebliebenen Positionen des Spektrums bestimmt. Da nach Satz 2 ein solcher Automat existiert, terminiert dieser Algorithmus immer. ■

**Korollar 6.1** *Für jede Realzeit-DCFL  $L$  sind  $\sigma^E(L)$ ,  $\sigma^Z(L)$  und  $\sigma^G(L)$  effektiv berechenbar.*

**Beweis** o. B. d. A. sei  $L$  gegeben durch einen DPDA  $A$  mit  $L(A) = L$ , der jedoch kein Realzeit-DPDA sein muß. Realzeit-DPDAs können in kanonischer Reihenfolge nach aufsteigender Größe aufgezählt werden. Für jeden solchen Realzeit-DPDA  $A_R$  ist entscheidbar, ob  $L(A_R) = L(A)$  gilt. Es gibt einen Realzeit-DPDA  $A_R$ , der  $L$  akzeptiert, und er wird mit diesem Verfahren in endlicher Zeit gefunden. Nun wird derselbe Algorithmus angewandt wie im Beweis von Satz 6, wobei nun jedoch kein Orakel benötigt wird, um für einen RDPDA  $A'$  zu entscheiden, ob  $L(A') = L(A_R) = L$  gilt. ■

# Kapitel 4

## Existenz von Verzögerungsspektren

In diesem Kapitel wird gezeigt, daß die meisten der in Abschnitt 3.2 angegebenen Schranken annähernd scharf sind. Es werden verschiedene Sprachfamilien definiert sowie obere und untere Schranken für ihre Verzögerungsspektren angegeben.

### 4.1 Einführende Ergebnisse und Definitionen

Zunächst wird eine obere Schranke für das Kellerwachstum von RDPDAs gezeigt.

**Lemma 5** *Sei  $A = (Q, \Sigma, \Gamma, \delta, C_0, F, q_r)$  ein RDPDA mit der Parsing-Eigenschaft. Dann gilt für jede Konfiguration  $C \in Q \times \Gamma^+$  und jedes Wort  $w \in \Sigma^+$*

$$|\delta_-(C, w)| < |C| + 2|w| \cdot |A|$$

**Beweis** Der Beweis wird als Widerspruchsbeweis geführt.

**Annahme:** Es gibt einen RDPDA  $A = (Q, \Sigma, \Gamma, \delta, C_0, F, q_r)$ , eine Konfiguration  $C \in Q \times \Gamma^+$  und ein Wort  $w \in \Sigma^+$ , so daß gilt

$$|\delta_-(C, w)| - |C| \geq 2|w| \cdot |A|$$

Die Berechnung  $C \xrightarrow{|w|} \delta_-(C, w)$  unterteilt sich in  $|w|$  Lese-Übergänge, und in  $|w|$  (eventuell leere) Folgen von  $\varepsilon$ -Übergängen. Die Länge des Kellerinhalts wächst in einem Lese-Übergang um weniger als  $|A|$  Symbole. Daraus folgt, daß der Kellerinhalt in den  $|w|$  Folgen von  $\varepsilon$ -Übergängen insgesamt um mehr als  $|w| \cdot |A|$  Symbole wächst. Demnach gibt es eine Berechnung

$$C_1 \xrightarrow{\varepsilon} C_2 \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} C_m$$

mit  $|C_m| > |C_1| + |A|$ .  $C_{i_1}, C_{i_2}, \dots, C_{i_l}$  seien alle Aufwärtskonfigurationen dieser Berechnung. Je zwei dieser Aufwärtskonfigurationen haben unterschiedliche Modi, da es sich sonst um Looping-Konfigurationen handeln würde. Außerdem gilt für jede Aufwärtskonfiguration  $C_{i_j}$ ,  $j \in [1 : l - 1]$ , daß die nächste Aufwärtskonfiguration  $C_{i_{j+1}}$  nicht länger ist als die unmittelbar auf  $C_{i_j}$  folgende Konfiguration  $C_{i_{j+1}}$ .

Daraus folgt für alle  $j \in [1 : l - 1]$

$$\begin{aligned} |C_{i_{j+1}}| &\leq |C_{i_{j+1}}| \\ &= |\delta(C_{i_j}, \varepsilon)| \\ &= |C_{i_j}| + \left| \delta\left(\left(C_{i_j}\right)^{[1]}, \varepsilon\right) \right| - 1 \end{aligned}$$

sowie

$$\begin{aligned} |C_m| &= |C_{i_l}| \\ &\leq |C_{i_1}| + \sum_{j=1}^{l-1} \left| \delta\left(\left(C_{i_j}\right)^{[1]}, \varepsilon\right) \right| \\ &\leq |C_1| + \sum_{\substack{M \in Q \times \Gamma \\ \delta(M, \varepsilon) \text{ ist definiert}}} |\delta(M, \varepsilon)| \\ &< |C_1| + |A| \end{aligned}$$

Durch diesen Widerspruch wird die Annahme widerlegt und das Lemma bewiesen. ■

Lemma 6 zeigt, daß ein RDPDA, der eine präfixfreie Sprache  $L$  (mit akzeptierenden Zuständen) erkennt, leicht dahingehend modifiziert werden kann, daß er  $L$  mit leerem Keller akzeptiert und fehlerhafte Eingaben, die nicht mit einem Wort aus  $L$  beginnen, weiterhin mit derselben Verzögerung zurückweist wie vorher.

**Lemma 6** Sei  $L$  eine Sprache, und sei  $A$  ein RDPDA, der  $L$  erkennt. Dann gibt es einen RDPDA  $A'$ , so daß gilt

1.  $N(A') = \min(L)$
2.  $A'$  weist inkorrekte Präfixe von  $L$ , die nicht mit einem Wort aus  $L$  beginnen, nach gleich vielen Übergängen desselben Typs zurück wie  $A$ .
3.  $|A'| \leq 3 \cdot |A|$

**Beweis** Sei  $A = (Q, \Sigma, \Gamma, \delta, (q_0, Z_0), F, q_r)$  ein RDPDA, der  $L$  erkennt.

Definiere

$$A' \stackrel{\text{def}}{=} ((Q - F) \cup \{q_f\}, \Sigma, \Gamma, \delta', (q'_0, Z_0), \emptyset, q_r),$$

wobei

$$q'_0 \stackrel{\text{def}}{=} \begin{cases} q_0, & \text{falls } \varepsilon \notin L \\ q_f, & \text{falls } \varepsilon \in L \end{cases}$$

$\delta'$  wird durch folgende Regeln definiert:

1. Falls  $\delta(q, Z, a) = (p, \gamma)$  mit  $p \in Q$  und  $\gamma \in \Gamma^*$  für ein  $q \in Q - F$ ,  $Z \in \Gamma$  und  $a \in \Sigma_\varepsilon$ , dann sei

$$\delta'(q, Z, a) \stackrel{\text{def}}{=} \begin{cases} (p, \gamma), & \text{falls } p \notin F, p \neq q_r \\ (q_f, \varepsilon), & \text{falls } p \in F \\ (q_r, Z), & \text{falls } p = q_r \end{cases}$$

2. Für alle  $Z \in \Sigma$  sei

$$\delta'(q_f, Z, \varepsilon) \stackrel{\text{def}}{=} (q_f, \varepsilon)$$

Mit Regel 1 simuliert  $A'$  die Übergänge auf nichtakzeptierenden Zuständen von  $A$ . Hierbei gibt es zwei Unterschiede gegenüber  $A$ : Zum einen sind alle akzeptierenden Zustände von  $A$  in  $q_f$  zusammengefaßt worden; zum anderen wird beim Übergang nach  $q_r$  immer ein Symbol auf den Keller geschrieben, damit  $A'$  hierbei nicht versehentlich den Keller leert. Wenn  $A$  in einen anderen Zustand als  $q_r$  übergeht, dann kann er ohnehin nicht seinen Keller leeren, da wegen der Parsing-Eigenschaft alle erreichbaren Stop-Konfigurationen von  $A$  zurückweisend sind. Regel 2 bewirkt, daß  $A'$  seinen Keller leert, sobald  $A$  einen akzeptierenden Zustand erreicht. Offensichtlich erfüllt  $A'$  die oben genannten Bedingungen. ■

Zum Abschluß dieses Abschnitts werden in Definition 17 einige Begriffe vereinbart, die in den folgenden Beweisen dieses Kapitels benötigt werden. Diese Begriffe werden bei ihrer ersten Verwendung auf Seite 51 mit Hilfe von Abbildung 4.1 erklärt.

**Definition 17** Sei  $A = (Q, \Sigma, \Gamma, \delta, C_0, F)$  ein DPDA.

Für drei Wörter  $x, y, z \in \Sigma^*$  wird  $x \cdot y \cdot z$  als **doppelt unterteiltes Wort** bezeichnet.  $x \cdot y$  ist eine kürzere Schreibweise für  $x \cdot y \cdot \varepsilon$ . In einem Kontext, in dem die Unterteilung ohne Bedeutung ist, ist  $x \cdot y \cdot z$  gleichbedeutend mit  $x y z$ .

Für jede Konfiguration  $C$  und jedes doppelt unterteilte Wort  $x \cdot y \cdot z$  mit  $\hat{\delta}(C, x y z) \neq \emptyset$  definiere die Konfiguration  $\delta_{\uparrow}(C, x \cdot y \cdot z)$  sowie die (nicht unterteilten) Wörter  $\text{begin}(C, x \cdot y \cdot z)$  und  $\text{mid} - \text{end}(C, x \cdot y \cdot z)$  so, daß gilt

- $\text{begin}(C, x \cdot y \cdot z) \text{mid} - \text{end}(C, x \cdot y \cdot z) = x y z$
- $C \mid \frac{\text{begin}(C, x \cdot y \cdot z)}{\quad} * \delta_{\uparrow}(C, x \cdot y \cdot z) \mid \frac{\text{mid} - \text{end}(C, x \cdot y \cdot z)}{\quad} * \delta_{+}(C, x y z)$
- $\delta_{\uparrow}(C, x \cdot y \cdot z)$  ist die erste Aufwärtskonfiguration (d. h., die letzte Konfiguration mit minimaler Länge) der Berechnung

$$\delta_{-}(C, x) \mid \frac{y}{\quad} * \delta_{-}(C, x y)$$

Falls  $|\delta_{\uparrow}(C, x y \cdot z)| \leq |\delta_{\uparrow}(C, x \cdot y \cdot z)|$  ist, dann definiere außerdem die Konfiguration  $\delta_{\downarrow}(C, x \cdot y \cdot z)$  sowie die Wörter  $\text{mid}(C, x \cdot y \cdot z)$  und  $\text{end}(C, x \cdot y \cdot z)$  so, daß gilt

- $\text{mid}(C, x \cdot y \cdot z) \text{end}(C, x \cdot y \cdot z) = \text{mid} - \text{end}(C, x \cdot y \cdot z)$

- $\delta_{\uparrow}(C, x \cdot y \cdot z) \stackrel{\text{mid}(C, x \cdot y \cdot z)}{\downarrow} \delta_{\downarrow}(C, x \cdot y \cdot z) \stackrel{\text{end}(C, x \cdot y \cdot z)}{\downarrow} \delta_{+}(C, x y z)$
- $\delta_{\downarrow}(C, x \cdot y \cdot z)$  ist die erste Konfiguration der Berechnung

$$\delta_{-}(C, x y) \stackrel{z}{\downarrow} \delta_{+}(C, x y z),$$

deren Länge derjenigen von  $\delta_{\uparrow}(C, x \cdot y \cdot z)$  entspricht.

$\text{begin} - \text{mid}(C, w)$  ist eine kürzere Schreibweise für das Wort  $\text{begin}(C, w) \text{mid}(C, w)$ .

## 4.2 Das Spektrum der Eingabeverzögerung

Offensichtlich sind alle Verzögerungsspektren monoton fallende Folgen; über die Spektren der Eingabeverzögerung ist aus Satz 2 außerdem bekannt, daß bei diesen höchstens exponentielle Größenunterschiede auftreten können. In Satz 7 wird gezeigt, daß dies im wesentlichen die einzige Beschränkung ist, der diese Spektren unterliegen.

**Satz 7** *Es gibt eine Konstante  $c \geq 1$ , so daß für jede Familie von monoton fallenden Funktionen  $(f_n)_{n \in \mathbb{N}}$  mit  $f_n : \mathbb{N}_0 \mapsto [1 : n]$  eine Familie von (regulären, präfixfreien) Sprachen  $(L'_{n, f_n})_{n \in \mathbb{N}}$  existiert, wobei für alle  $n \geq 2$  gilt*

$$\sigma^E(L'_{n, f_n}) \in \left( \begin{array}{c} n^{c \cdot f_n(0)} + n^c \cdot |f_n| \\ n^{f_n(0)} \end{array} \middle| \dots \middle| \begin{array}{c} n^{c \cdot f_n(d)} + n^c \cdot |f_n| \\ n^{f_n(d)} \end{array} \middle| \dots \middle| \begin{array}{c} n^c \cdot |f_n| \\ n \end{array} \right)$$

**Beweis** Definiere für alle  $n \in \mathbb{N}$  und jede monoton fallende Funktion  $f_n : \mathbb{N}_0 \mapsto [1 : n]$

$$\begin{aligned} \Sigma_n &\stackrel{\text{def}}{=} [1 : n] \\ \Sigma'_n &\stackrel{\text{def}}{=} [1 : n] \times [1 : n] \\ L_{n, f_n} &\stackrel{\text{def}}{=} \left\{ a_1 \cdots a_n [i, b] v [j, c] \mid a_1, \dots, a_n \in \Sigma_n, v \in \Sigma_n^*, [i, b], [j, c] \in \Sigma'_n, \right. \\ &\quad \left. a_i = b, a_j = c, j \leq f_n(|v|) \right\} \\ L'_{n, f_n} &\stackrel{\text{def}}{=} L_{4n+12, 4f_n+12}^\dagger \end{aligned}$$

Die Unterscheidung zwischen  $L_{n,f_n}$  und  $L'_{n,f_n}$  ermöglicht eine einfachere Formulierung des folgenden Beweises. Demselben Zweck dient die Erweiterung der Funktion  $f_n$  auf den Definitionsbereich  $IN_{0,\infty}$  durch die Definition  $f_n(\infty) \stackrel{\text{def}}{=} 0$ .

**1. Teil:** Zu zeigen:

*Für jeden RDPDA  $A$ , der  $L'_{n,f_n}$  mit Eingabeverzögerung  $d$  für ein  $n \in IN$  und  $d \in IN_{0,\infty}$  erkennt, gilt*

$$|A| \geq \max \{n, n^{f_n(d)}\}$$

Zunächst soll die notwendige Intuition vermittelt werden, um den folgenden formalen Beweis leichter zu verstehen. Anstatt die obige Behauptung für  $L'_{n,f_n}$  direkt zu beweisen, wird zunächst gezeigt, daß jeder RDPDA, der  $L_{n,f_n}$  mit Eingabeverzögerung  $d$  erkennt, eine bestimmte Größe nicht unterschreiten kann. Der Fall  $d = \infty$  ist trivial, denn für jeden RDPDA  $A$ , der  $L_{n,f_n}$  erkennt, gilt  $|A| \geq n^2$ , da  $A$  sonst nicht alle Eingabezeichen lesen könnte. Somit ist nur noch der Fall  $d \in IN_0$  zu betrachten.

Sei  $A$  ein RDPDA, der  $L_{n,f_n}$  mit Eingabeverzögerung  $d$  für ein  $d \in IN_0$  erkennt.  $A$  werde auf der Eingabe  $a_1 \cdots a_n [i, b] v [j, c]$  mit  $|v| = d$  und  $i < f_n(d)$  gestartet und habe bereits  $a_1 \cdots a_n [i, b] v$  gelesen.  $A$  muß feststellen, ob  $a_1 \cdots a_n [i, b]$  ein korrektes Präfix von  $L_{n,f_n}$  ist, bevor er das letzte Zeichen  $[j, c]$  liest. Hierzu muß  $A$  auf  $a_i$  zugreifen, ohne jedoch  $a_1 \cdots a_{f_n(d)}$  zu löschen, da er sonst  $a_j = c$  nicht in jedem Fall entscheiden könnte. Daraus folgt, daß  $a_1 \cdots a_{f_n(d)}$  in einem 1-Modus von  $A$  zu speichern sein muß, und dies ist der bestimmende Faktor für die Größe eines minimalen RDPDA, der  $L_{n,f_n}$  mit Eingabeverzögerung  $d$  erkennt.

**Annahme:** Es gibt einen RDPDA  $A$  der Größe  $|A| < n^{\lfloor f_n(d)/4 \rfloor} / (6 f_n(d))$ , der  $L_{n,f_n}$  mit Eingabeverzögerung  $d$  für ein  $n \in IN$  und  $d \in IN_0$  erkennt.

---

<sup>†</sup>Eine arithmetische Operation, angewandt auf eine Funktion  $f$ , meint die Anwendung der Operation auf jeden Funktionswert von  $f$ .

o. B. d. A. sei  $f_n(d)$  ein Vielfaches von 4. Diese Annahme ist zulässig, da letztlich nicht  $L_{n,f_n}$ , sondern  $L'_{n,f_n} = L_{4n+12,4f_n+12}$  interessiert. Konstruiere, ausgehend von  $A$ , gemäß Lemma 6 auf Seite 47 einen RDPDA

$$A' = (Q, \Sigma_n \cup \Sigma'_n, \Gamma, \delta, C_0, \{q_f\}, q_r)$$

mit  $N(A') = L_{n,f_n}$ . Für bestimmte Eingabewörter  $w \in N(A')$  wird die akzeptierende Berechnung von  $A'$  auf der Eingabe  $w$  in drei Abschnitte aufgeteilt, die eine Zerlegung von  $w$  in drei Teilwörter induzieren. Es werden zwei Wörter  $w_1, w_2 \in N(A')$  konstruiert, so daß ein Teilwort von  $w_1$  gegen das korrespondierende Teilwort von  $w_2$  ausgetauscht werden kann und das entstehende Wort  $w_0$  ebenfalls von  $A'$  akzeptiert wird, obwohl es nicht in  $L_{n,f_n}$  ist.

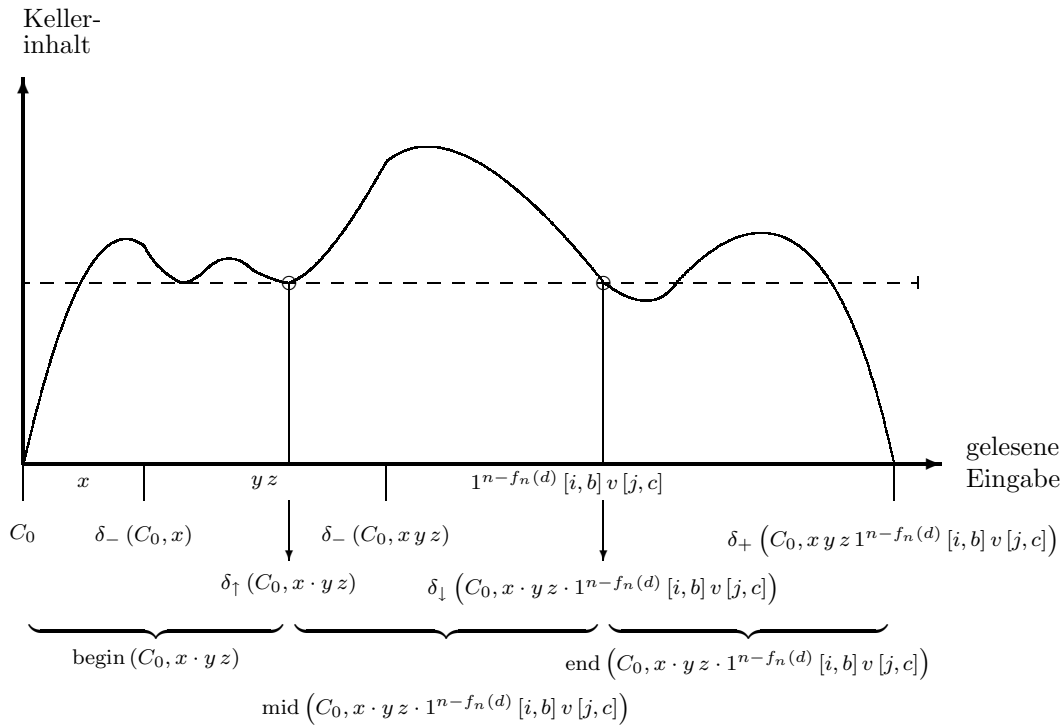


Abbildung 4.1: Akzeptierende Berechnung von  $A'$

Für jedes Wort  $xy z 1^{n-f_n(d)}[i,b]v[j,c] \in L_{n,f_n}$  mit  $x, z \in \Sigma_n^{f_n(d)/4}$ ,  $y \in \Sigma_n^{f_n(d)/2}$ ,  $v \in \Sigma_n^*$  und  $[i,b], [j,c] \in \Sigma'_n$  läßt sich die zugehörige Berechnung so aufteilen, wie es in Abbildung 4.1 dargestellt ist. Nun werden die Wortpaare  $(x, y) \in \Sigma_n^{f_n(d)/4} \times \Sigma_n^{f_n(d)/2}$  in Abhängigkeit davon, ob sich  $xy$  durch ein Wort  $z \in \Sigma_n^{f_n(d)/4}$  so fortsetzen läßt, daß das Teilwort  $\text{begin}(C_0, x \cdot yz)$  mindestens so lang ist wie  $xy$ , in zwei Klassen eingeteilt.

Definiere

$$\begin{aligned}\mathcal{U} &\stackrel{\text{def}}{=} \left\{ (x, y) \in \Sigma_n^{f_n(d)/4} \times \Sigma_n^{f_n(d)/2} \mid \exists z \in \Sigma_n^{f_n(d)/4} : |\text{begin}(C_0, x \cdot y z)| \geq |x y| \right\} \\ \mathcal{U}' &\stackrel{\text{def}}{=} \left\{ (x, y) \in \Sigma_n^{f_n(d)/4} \times \Sigma_n^{f_n(d)/2} \mid \forall z \in \Sigma_n^{f_n(d)/4} : |\text{begin}(C_0, x \cdot y z)| < |x y| \right\}\end{aligned}$$

### Fallunterscheidung

**Fall 1:**  $\#\mathcal{U} \geq \#\mathcal{U}'$

Definiere für alle  $x \in \Sigma_n^{f_n(d)/4}$

$$\mathcal{V}(x) \stackrel{\text{def}}{=} \left\{ y \in \Sigma_n^{f_n(d)/2} \mid (x, y) \in \mathcal{U} \right\}$$

und wähle  $x_0$  so, daß  $\mathcal{V}(x_0)$  maximale Kardinalität hat.

Für alle  $y \in \mathcal{V}(x_0)$  sei  $z_y$  das lexikographisch kleinste Wort aus  $\Sigma_n^{f_n(d)/2}$ , so daß gilt

$$|\text{begin}(C_0, x_0 \cdot y z_y)| \geq |x_0 y|$$

Aus Abbildung 4.1 ist ersichtlich, daß der Kellerinhalt von  $\delta_{\uparrow}(C_0, x_0 \cdot y z_y)$  ohne das oberste Kellersymbol für alle  $y \in \mathcal{V}(x_0)$  ein echtes Präfix des Kellerinhalts von  $\delta_{-}(C_0, x_0)$  ist, so daß die Anzahl der Konfigurationen von der Form  $\delta_{\uparrow}(C_0, x_0 \cdot y z_y)$  durch das Produkt der Länge von  $\delta_{-}(C_0, x_0)$  und der Anzahl der fortsetzbaren Modi von  $A'$  beschränkt ist. Wie im folgenden gezeigt wird, reicht die Anzahl dieser Konfigurationen nicht aus, um all diejenigen Präfixe von  $y z_y$  zu unterscheiden, die während der Berechnung von  $\delta_{-}(C_0, x_0)$  nach  $\delta_{\uparrow}(C_0, x_0 \cdot y z_y)$  gelesen werden.

Es gilt

$$\begin{aligned}\#\mathcal{V}(x_0) &\geq \#\mathcal{U} / \#\Sigma_n^{f_n(d)/4} \\ &\geq n^{f_n(d)/2} / 2 \\ &> f_n(d) \cdot |A'|^2 \\ &\geq \#(A')^{[1]} \cdot |\delta_{-}(C_0, x_0)| \quad (\text{nach Lemma 5}) \\ &\geq \#\left\{ \delta_{\uparrow}(C_0, x_0 \cdot y z_y) \mid y \in \mathcal{V}(x_0) \right\}\end{aligned}$$

Daraus folgt

$$\begin{aligned} \exists y_1, y_2 \in \mathcal{V}(x_0), i_0 \in [f_n(d)/4 + 1 : 3f_n(d)/4], b_1, b_2 \in \Sigma_n : \\ b_1 = (y_1)_{[i_0 - f_n(d)/4]} \neq (y_2)_{[i_0 - f_n(d)/4]} = b_2, \\ \delta_{\uparrow}(C_0, x_0 \cdot y_1 z_{y_1}) = \delta_{\uparrow}(C_0, x_0 \cdot y_2 z_{y_2}) \end{aligned}$$

Definiere

$$\begin{aligned} w_1 &\stackrel{\text{def}}{=} x_0 \cdot y_1 z_{y_1} \cdot 1^{n-f_n(d)} [i_0, b_1] [i_0, b_1] \\ w_2 &\stackrel{\text{def}}{=} x_0 \cdot y_2 z_{y_2} \cdot 1^{n-f_n(d)} [i_0, b_2] [i_0, b_2] \\ w_0 &\stackrel{\text{def}}{=} \text{begin}(C_0, w_2) \text{mid} - \text{end}(C_0, w_1) \end{aligned}$$

Es gilt  $w_1, w_2 \in L_{n, f_n}$ , aber  $w_0 \notin L_{n, f_n}$ , da  $(y_2)_{[i_0 - f_n(d)/4]} \neq b_1$  ist. Wegen  $\delta_{\uparrow}(C_0, w_1) = \delta_{\uparrow}(C_0, w_2)$  kann  $A'$  die Wörter  $w_0$  und  $w_1$  jedoch nicht unterscheiden und akzeptiert somit entweder keines dieser Wörter oder beide. Dies steht im Widerspruch zur Annahme,  $A'$  akzeptiere  $L_{n, f_n}$ .

**Fall 2:**  $\#\mathcal{U} < \#\mathcal{U}'$

Definiere für alle  $y \in \Sigma_n^{f_n(d)/2}$

$$\mathcal{V}'(y) \stackrel{\text{def}}{=} \left\{ x \in \Sigma_n^{f_n(d)/4} \mid (x, y) \in \mathcal{U}' \right\}$$

und wähle  $y_0$  so, daß  $\mathcal{V}'(y_0)$  maximale Kardinalität hat.

Es gilt

$$\begin{aligned} \#\mathcal{V}'(y_0) &\geq \#\mathcal{U}' / \#\Sigma_n^{f_n(d)/2} \\ &\geq n^{f_n(d)/4} / 2 \\ &> \#(A')^{[1]} \\ &\geq \#\left\{ (\delta_{\uparrow}(C_0, x \cdot y_0))^{[1]} \mid x \in \mathcal{V}'(y_0) \right\} \end{aligned}$$

Daraus folgt

$$\begin{aligned} \exists x_1, x_2 \in \mathcal{V}'(y_0), i_0 \in [1 : f_n(d)/4], b_1, b_2 \in \Sigma_n : \\ b_1 = (x_1)_{[i_0]} \neq (x_2)_{[i_0]} = b_2, \\ (\delta_{\uparrow}(C_0, x_1 \cdot y_0))^{[1]} = (\delta_{\uparrow}(C_0, x_2 \cdot y_0))^{[1]} \end{aligned}$$

Nun wird eine weitere Fallunterscheidung vorgenommen, abhängig davon, ob es ein Wort  $z \in \Sigma_n^{f_n(d)/4}$  gibt, so daß gilt

$$\left| \text{end} \left( C_0, x_1 \cdot y_0 z \cdot 1^{n-f_n(d)} [i_0, b_1] 1^d [i_0, b_1] \right) \right| = 0$$

### Fallunterscheidung

**Fall 2.1:**  $\forall z \in \Sigma_n^{f_n(d)/4} : \left| \text{end} \left( C_0, x_1 \cdot y_0 z \cdot 1^{n-f_n(d)} [i_0, b_1] 1^d [i_0, b_1] \right) \right| > 0$

Wegen  $\#(A')^{[1]} < n^{f_n(d)/4}$  gilt

$$\begin{aligned} \exists z_1, z_2 \in \Sigma_n^{f_n(d)/4}, j_0 \in [3f_n(d)/4 + 1 : f_n(d)], c_1, c_2 \in \Sigma_n : \\ c_1 = (z_1)_{[j_0-3f_n(d)/4]} \neq (z_2)_{[j_0-3f_n(d)/4]} = c_2, \\ \left( \delta_{\downarrow} \left( C_0, x_1 \cdot y_0 z_1 \cdot 1^{n-f_n(d)} [i_0, b_1] 1^d [i_0, b_1] \right) \right)^{[1]} = \\ \left( \delta_{\downarrow} \left( C_0, x_1 \cdot y_0 z_2 \cdot 1^{n-f_n(d)} [i_0, b_1] 1^d [i_0, b_1] \right) \right)^{[1]} \end{aligned}$$

Definiere

$$\begin{aligned} w_1 &\stackrel{\text{def}}{=} x_1 \cdot y_0 z_1 \cdot 1^{n-f_n(d)} [i_0, b_1] 1^d [j_0, c_1] \\ w_2 &\stackrel{\text{def}}{=} x_1 \cdot y_0 z_2 \cdot 1^{n-f_n(d)} [i_0, b_1] 1^d [j_0, c_2] \\ w_0 &\stackrel{\text{def}}{=} \text{begin}(C_0, w_1) \text{ mid}(C_0, w_2) \text{ end}(C_0, w_1) \end{aligned}$$

Es gilt  $w_1, w_2 \in L_{n, f_n}$ , aber  $w_0 \notin L_{n, f_n}$ , da  $|\text{end}(C_0, w_1)| > 0$  und  $(z_2)_{[j_0-3f_n(d)/4]} \neq c_1$  ist.  $A'$  kann die Wörter  $w_1$  und  $w_0$  jedoch nicht unterscheiden und akzeptiert somit entweder keines dieser Wörter oder beide. Dies steht im Widerspruch zur Annahme,  $A'$  akzeptiere  $L_{n, f_n}$ .

**Fall 2.2:**  $\exists z_0 \in \Sigma_n^{f_n(d)/4} : \left| \text{end} \left( C_0, x_1 \cdot y_0 z_0 \cdot 1^{n-f_n(d)} [i_0, b_1] 1^d [i_0, b_1] \right) \right| = 0$

Definiere

$$w_1 \stackrel{\text{def}}{=} x_1 \cdot y_0 z_0 \cdot 1^{n-f_n(d)} [i_0, b_1] 1^d [i_0, b_1]$$

$$w_2 \stackrel{\text{def}}{=} x_2 \cdot y_0 z_0 \cdot 1^{n-f_n(d)} [i_0, b_2] 1^d [i_0, b_2]$$

$$w_0 \stackrel{\text{def}}{=} \text{begin} (C_0, w_2) \text{ mid} (C_0, w_1)$$

Es gilt  $w_1, w_2 \in L_{n, f_n}$ , hingegen ist  $w_0^{-[d+1]}$  ein inkorrektes Präfix von  $L_{n, f_n}$ , da  $|\text{end}(C_0, w_1)| = 0$  und  $(x_2)_{[i_0]} \neq b_1$  ist. Wegen  $(\delta_{\uparrow}(C_0, w_1))^{[1]} = (\delta_{\uparrow}(C_0, w_2))^{[1]}$  wird  $w_0$  von  $A'$  (und von  $A$ ) jedoch vollständig gelesen. Dies steht im Widerspruch zur Annahme,  $A$  erkenne  $L_{n, f_n}$  mit Eingabeverzögerung  $d$ .

Alle Fälle führen zu einem Widerspruch. Somit ist die Annahme,  $A$  erkenne  $L_{n, f_n}$  mit Eingabeverzögerung  $d$ , widerlegt.

Daraus folgt

$$\begin{aligned} \sigma_d^E(L'_{n, f_n}) &= \sigma_d^E(L_{4n+12, 4f_n+12}) \\ &\geq \frac{(4n+12)^{f_n(d)+3}}{6 \cdot (4f_n(d)+12)} \\ &\geq \frac{(4n+12)^3}{24n+72} \cdot n^{f_n(d)} \\ &\geq n^{f_n(d)} \end{aligned}$$

**2. Teil:** Zu zeigen:

*Es gibt eine Konstante  $c > 0$ , so daß für alle  $n \geq 2$  und  $d \in [0 : |f_n|] \cup \{\infty\}$  ein moderater RDPDA  $A$  der Größe  $|A| \leq n^{c \cdot f_n(d)} + n^c \cdot |f_n|$  existiert, der  $L_{n, f_n}$  mit Eingabeverzögerung  $d$  erkennt.*

Definiere

$$A \stackrel{\text{def}}{=} (Q, \Sigma_n \cup \Sigma'_n, \Sigma'_n \cup \{Z_0\}, \delta, ([\uparrow, 0, Z_0], Z_0), \{q_f\}, q_r),$$

wobei

$$\begin{aligned} Q \stackrel{\text{def}}{=} & \left\{ [\uparrow, k, x] \mid \left( k \in [0 : f_n(d) - 1], x \in Z_0 \cdot (\Sigma'_n)^k \right) \vee \right. \\ & \left( k = f_n(d), x \in Z_0 \cdot (\Sigma'_n)^{\leq k} \right) \vee \\ & \left. \left( k \in [f_n(d) : n], x = \varepsilon \right) \right\} \cup \\ & \left\{ [\downarrow, l, s] \mid l \in [0 : |f_n|], s \in \Sigma'_n \right\} \cup \\ & \left\{ [\downarrow, s, t] \mid s, t \in \Sigma'_n \right\} \cup \{q_f, q_r\} \end{aligned}$$

sowie für alle  $k \in [0 : n]$ ,  $l \in [0 : |f_n|]$ ,  $x \in (\Sigma'_n \cup \{Z_0\})^*$ ,  $s, t \in \Sigma'_n$ ,  $Z \in \Sigma'_n \cup \{Z_0\}$  und  $a \in \Sigma_n \cup \Sigma'_n$

$$\delta([\uparrow, k, x], Z, a) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} ([\uparrow, k+1, x[k+1, a]], Z[k+1, a]), \\ \quad \text{falls } k \in [0 : f_n(d) - 1], a \in \Sigma_n \\ ([\uparrow, k, {}^{-[1]}x], Z x_{[1]}), a = \varepsilon, \\ \quad \text{falls } k = f_n(d), |x| > 0 \\ ([\uparrow, k+1, \varepsilon], Z[k+1, a]), \\ \quad \text{falls } k \in [f_n(d) : n-1], |x| = 0, a \in \Sigma_n \\ ([\downarrow, 0, a], Z Z), \\ \quad \text{falls } k = n, |x| = 0, a \in \Sigma'_n \\ (q_r, \varepsilon), \\ \quad \text{falls } (k \in [0 : f_n(d) - 1], a \in \Sigma'_n) \vee \\ \quad (k \in [f_n(d) : n-1], |x| = 0, a \in \Sigma'_n) \vee \\ \quad (k = n, |x| = 0, a \in \Sigma_n) \end{array} \right.$$

$$\begin{aligned}
\delta([\downarrow, l, s], Z, a) &\stackrel{\text{def}}{=} \left\{ \begin{array}{l} ([\downarrow, \min\{l+1, |f_n|\}, s], Z), \\ \text{falls } (l \neq d \vee Z = Z_0), a \in \Sigma_n \\ ([\downarrow, s, a], \varepsilon), \\ \text{falls } (l \neq d \vee Z = Z_0), a \in \Sigma'_n, \text{first}(a) \leq f_n(l) \\ (q_r, \varepsilon), \\ \text{falls } (l \neq d \vee Z = Z_0), a \in \Sigma'_n, \text{first}(a) > f_n(l) \\ ([\downarrow, l, s], \varepsilon), a = \varepsilon, \\ \text{falls } l = d, Z \in \Sigma'_n, \\ \quad (\text{first}(Z) \neq \text{first}(s) \vee \text{second}(Z) = \text{second}(s)) \\ (q_r, \varepsilon), a = \varepsilon, \\ \text{falls } l = d, Z \in \Sigma'_n, \\ \quad (\text{first}(Z) = \text{first}(s), \text{second}(Z) \neq \text{second}(s)) \end{array} \right. \\
\delta([\downarrow, s, t], Z, \varepsilon) &\stackrel{\text{def}}{=} \left\{ \begin{array}{l} ([\downarrow, s, t], \varepsilon), \\ \text{falls } Z \in \Sigma'_n, \\ \quad (\text{first}(Z) \neq \text{first}(s) \vee \text{second}(Z) = \text{second}(s)), \\ \quad (\text{first}(Z) \neq \text{first}(t) \vee \text{second}(Z) = \text{second}(t)) \\ (q_r, \varepsilon), \\ \text{falls } Z \in \Sigma'_n, \\ \quad (\text{first}(Z) = \text{first}(s), \text{second}(Z) \neq \text{second}(s)) \vee \\ \quad (\text{first}(Z) = \text{first}(t), \text{second}(Z) \neq \text{second}(t)) \\ (q_f, Z_0), \\ \text{falls } Z = Z_0 \end{array} \right. \\
\delta(q_f, Z_0, a) &\stackrel{\text{def}}{=} (q_r, \varepsilon)
\end{aligned}$$

$A$  verarbeitet eine Eingabe  $a_1 \cdots a_n [i, b] v [j, c]$  folgendermaßen: Anfangs befindet sich  $A$  in Zuständen der Form  $[\uparrow, k, x]$ , wobei  $k$  die Anzahl der bisher gelesenen Eingabezeichen angibt.  $A$  liest zunächst  $a_1 \cdots a_{f_n(d)}$ , schreibt diese Zeichen zusammen mit ihrer jeweiligen Position („1“ für  $a_1$ , „2“ für  $a_2$  usw.) auf den Keller und speichert sie vorübergehend

in der endlichen Kontrolle, um sie dann in Zuständen der Form  $[\uparrow, f_n(d), x]$  zusammen mit dem Trennsymbol  $Z_0$  ein zweites Mal auf den Keller zu schreiben. Anschließend werden  $a_{f_n(d)+1} \cdots a_n$  gelesen und auf den Keller geschrieben, so daß dieser schließlich  $Z_0 [1, a_1] \cdots [f_n(d), a_{f_n(d)}] Z_0 [1, a_1] \cdots [n, a_n]$  enthält. Nun geht  $A$  mit dem Zeichen  $[i, b]$  in den Zustand  $[\downarrow, 0, [i, b]]$  über, wobei er aus technischen Gründen das oberste Kellersymbol dupliziert.

In Zuständen der Form  $[\downarrow, l, [i, b]]$  wird  $v$  gelesen, wobei  $l$  die Anzahl der bisher gelesenen Zeichen angibt. Falls  $|v| \geq d$  ist, dann testet  $A$  nach dem Lesen von  ${}^{[d]}v$  zunächst durch Leeren des Kellers bis zum Symbol  $Z_0$ , ob  $a_i = b$  gilt; dieser Test unterbleibt im Fall  $|v| < d$ . Nachdem  $A$  alle Zeichen aus  $\Sigma_n$  gelesen hat, geht er mit dem Zeichen  $[j, c] \in \Sigma'_n$  in den Zustand  $[\Downarrow, [i, b], [j, c]]$  über, falls  $j \leq f_n(|v|)$  gilt.

In Zuständen der Form  $[\Downarrow, [i, b], [j, c]]$  leert  $A$  den Keller mit  $\varepsilon$ -Übergängen bis zum nächsten Auftreten von  $Z_0$ . Dabei testet er, ob  $a_j = c$  gilt und zumindest im Fall  $|v| < d$  auch, ob  $a_i = b$  gilt.

Offensichtlich erkennt  $A$  die Sprache  $L_{n, f_n}$ . Nun ist noch zu betrachten, mit welcher Eingabeverzögerung  $A$  fehlerhafte Eingaben zurückweist. Minimale inkorrekte Präfixe der Form  $a_1 \cdots a_n [i, b]$  mit  $a_i \neq b$  werden nach dem Lesen von höchstens  $d$  weiteren Zeichen zurückgewiesen; minimale inkorrekte Präfixe der Form  $a_1 \cdots a_n [i, b] v [j, c]$  mit  $j > f_n(|v|)$  oder  $a_j \neq c$  werden zurückgewiesen, ohne ein weiteres Zeichen zu lesen. Daraus folgt, daß  $A$  die Sprache  $L_{n, f_n}$  mit Eingabeverzögerung  $d$  erkennt. Die Größe von  $A$  beträgt

$$|A| = O\left(n^{f_n(d)+3} + n^6 \cdot |f_n|\right)$$

■

Die folgenden Korollare verdeutlichen die Mächtigkeit von Satz 7, indem sie einige Sprachfamilien mit besonders interessanten Spektren der Eingabeverzögerung angeben.

**Korollar 7.1** *Es gibt eine Familie von (regulären, präfixfreien) Sprachen  $(L'_n)_{n \in \mathbb{N}}$  und eine Konstante  $c \geq 1$ , so daß für alle  $n \geq 2$  gilt*

$$\sigma^E(L'_n) \in \left( \begin{array}{c} | \\ n^n \\ | \\ n^n \\ | \\ \cdots \\ | \\ n^n \\ | \\ \cdots \\ | \\ n^c \end{array} \right)$$

**Beweis** Definiere für alle  $n \in \mathbb{N}$  die konstante Funktion  $f_n : \mathbb{N}_0 \mapsto \{n\}, k \rightarrow n$ , und wähle  $L'_n$  als die Sprache  $L'_{n,f_n}$  aus dem Beweis von Satz 7. ■

Korollar 7.1 stellt eine Erweiterung von [8, Theorem 4.5] dar. Dort wurde — unter Beschränkung auf Scanning-DPDAs, eine Teilklasse der moderaten DPDAs — gezeigt, daß es eine Sprachfamilie  $(E_n)_{n \in \mathbb{N}}$  gibt, so daß die Produkt-Größe von minimalen Correct-Prefix-Parsern für  $E_n$  exponentiell in  $n$  wächst, während die Produkt-Größe von beliebigen Parsern für  $E_n$  lediglich kubisch wächst.

**Korollar 7.2** Für alle  $m \in \mathbb{N}$  gibt es eine Familie von (regulären, präfixfreien) Sprachen  $(L'_{m,n})_{n \in \mathbb{N}}$  und eine Konstante  $c \geq 1$ , so daß für alle  $n \geq 2$  gilt

$$\sigma^E(L'_{m,n}) \in \left( \underbrace{\left( n^n \mid n^n \mid \cdots \mid n^n \right)}_{m \text{ mal}} \mid mn^c \mid mn^c \mid \cdots \parallel mn^c \right)$$

**Beweis** Definiere für alle  $m, n \in \mathbb{N}$  die Funktion  $f_{m,n} : \mathbb{N}_0 \mapsto [1 : n]$  durch

$$f_{m,n}(k) \stackrel{\text{def}}{=} \begin{cases} n, & \text{falls } k < m \\ 1, & \text{falls } k \geq m \end{cases}$$

und wähle  $L'_{m,n}$  als die Sprache  $L'_{n,f_{m,n}}$  aus dem Beweis von Satz 7. ■

**Korollar 7.3** Es gibt eine Familie von (regulären, präfixfreien) Sprachen  $(L'_n)_{n \in \mathbb{N}}$  und eine Konstante  $c \geq 1$ , so daß für alle  $n \geq 2$  gilt

$$\sigma^E(L'_n) \in \left( n^{cn} \mid n^{c(n-1)} \mid n^{c(n-2)} \mid \cdots \mid n^c \mid n^c \mid \cdots \parallel n^c \right)$$

**Beweis** Definiere für alle  $n \in \mathbb{N}$  die Funktion  $f_n : \mathbb{N}_0 \mapsto [1 : n]$  durch

$$f_n(k) \stackrel{\text{def}}{=} \begin{cases} n - k, & \text{falls } k < n \\ 1, & \text{falls } k \geq n \end{cases}$$

und wähle  $L'_n$  als die Sprache  $L'_{n,f_n}$  aus dem Beweis von Satz 7. ■

Korollar 7.3 entspricht weitgehend einem Ergebnis aus [16].

### 4.3 Das Spektrum der Zeitverzögerung

**Satz 8** *Es gibt eine Familie von (regulären, präfixfreien) Sprachen  $(L'_n)_{n \in \mathbb{N}}$  und eine Konstante  $c \geq 1$ , so daß für alle  $n \geq 2$  gilt*

$$\sigma^Z(L'_n) \in \left( n^n \mid n^n \mid \cdots \mid n^n \mid \cdots \parallel n^c \right)$$

**Beweis** Definiere für alle  $n \in \mathbb{N}$

$$\Sigma_n \stackrel{\text{def}}{=} [-n : -1] \cup [1 : n]$$

$$\Sigma'_n \stackrel{\text{def}}{=} [-n : n]$$

$$L_n \stackrel{\text{def}}{=} \{x a y 0^+ a \mid x \in \Sigma_n^*, a \in \Sigma_n, y \in (\Sigma_n - \{a, -a\})^*\}$$

$$L'_n \stackrel{\text{def}}{=} L_{6 \lceil \log_2 n + 1 \rceil \cdot (n+5)}$$

**1. Teil:** Zu zeigen:

*Für jeden RDPDA  $A$ , der  $L'_n$  mit Zeitverzögerung  $d$  für ein  $n \geq 2$  und  $d \in \mathbb{N}_0$  erkennt, gilt*

$$|A| \geq n^n$$

Anstatt die obige Behauptung für  $L'_n$  direkt zu beweisen, wird zunächst gezeigt, daß jeder RDPDA, der  $L_n$  mit endlicher Zeitverzögerung erkennt, eine bestimmte Größe nicht unterschreiten kann. Dieser Teil des Beweises verläuft in weiten Teilen analog zum 1. Teil des Beweises von Satz 7.

**Annahme:** Es gibt einen RDPDA  $A$  der Größe  $|A| < 2^{\lfloor n/6 \rfloor} / (2n)$ , der  $L_n$  mit Zeitverzögerung  $d$  für ein  $n \in \mathbb{N}$  und  $d \in \mathbb{N}_0$  erkennt.

o. B. d. A. sei  $n$  ein Vielfaches von 3. Diese Annahme ist zulässig, da letztlich nicht das Erkennen von  $L_n$ , sondern von  $L'_n = L_{6 \lceil \log_2 n + 1 \rceil \cdot (n+5)}$  interessiert. Die Beweisidee ist folgende: Es werden zwei Wörter  $w \in L_n$  und  $w' \in MIP(L_n)$  definiert, so daß die  $d$ -Modi von  $A$

nach dem Lesen von  $w$  und  $w'$  übereinstimmen. Demnach muß  $A$  nach dem Lesen von  $w'$  mehr als  $d$  Übergänge machen, bis er diese Eingabe von  $w$  unterscheiden und zurückweisen kann. Dies steht im Widerspruch zur Annahme,  $A$  erkenne  $L_n$  mit Zeitverzögerung  $d$ .

Sei  $A = (Q, \Sigma'_n, \Gamma, \delta, C_0, F, q_r)$ . Zunächst werden zwei Wörter  $v$  und  $v'$  konstruiert, so daß  $v \not\equiv_{L_n} v'$  ist und  $C_0 \stackrel{v}{\vdash}^* C_1$  sowie  $C_0 \stackrel{v'}{\vdash}^* C'_1$  gilt mit  $(C_1)^{[1]} = (C'_1)^{[1]}$ .

Definiere die Wörter

$$\begin{aligned} u_0 &\stackrel{\text{def}}{=} \langle\langle[-n/3 : -1]\rangle\rangle \\ u_1 &\stackrel{\text{def}}{=} \langle\langle[-2n/3 : -(n/3 + 1)]\rangle\rangle \end{aligned}$$

sowie die Sprachen

$$\begin{aligned} \mathcal{T}_0 &\stackrel{\text{def}}{=} \{ \langle\langle S \rangle\rangle \mid S \subseteq [1 : n/3] \} \\ \mathcal{T}_1 &\stackrel{\text{def}}{=} \{ \langle\langle S \rangle\rangle \mid S \subseteq [n/3 + 1 : 2n/3] \} \\ \mathcal{T}_2 &\stackrel{\text{def}}{=} \{ \langle\langle S \rangle\rangle \mid S \subseteq [2n/3 + 1 : n] \} \end{aligned}$$

und die Mengen von Wortpaaren

$$\begin{aligned} \mathcal{U} &\stackrel{\text{def}}{=} \{ (x, y) \in \mathcal{T}_2 \times \mathcal{T}_0 \mid \exists z \in \mathcal{T}_1 : |\text{begin}(C_0, x \cdot y u_1 z 0 0)| \geq |x y| \} \\ \mathcal{U}' &\stackrel{\text{def}}{=} \{ (x, y) \in \mathcal{T}_2 \times \mathcal{T}_0 \mid \forall z \in \mathcal{T}_1 : |\text{begin}(C_0, x \cdot y u_1 z 0 0)| < |x y| \} \end{aligned}$$

### Fallunterscheidung

**Fall 1:**  $\#\mathcal{U} \geq \#\mathcal{U}'$

Definiere für alle  $x \in \mathcal{T}_2$

$$\mathcal{V}(x) \stackrel{\text{def}}{=} \{ y \in \mathcal{T}_0 \mid (x, y) \in \mathcal{U} \}$$

und wähle  $x_0$  so, daß  $\mathcal{V}(x_0)$  maximale Kardinalität hat.

Für alle  $y \in \mathcal{V}(x_0)$  sei  $z_y$  das lexikographisch kleinste Wort aus  $\mathcal{T}_1$ , so daß gilt

$$|\text{begin}(C_0, x_0 \cdot y u_1 z_y 0 0)| \geq |x_0 y|$$

Da der Kellerinhalt von  $\delta_{\uparrow}(C_0, x_0 \cdot y u_1 z_y 0 0)$  ohne das oberste Kellersymbol für alle  $y \in \mathcal{V}(x_0)$  ein echtes Präfix des Kellerinhalts von  $\delta_{-}(C_0, x_0)$  ist, gilt

$$\begin{aligned}
\#\mathcal{V}(x_0) &\geq \#\mathcal{U}/\#\mathcal{T}_2 \\
&\geq \#\mathcal{T}_0/2 \\
&= 2^{n/3}/2 \\
&> 2^{\frac{n}{3}} \cdot |A|^2 \\
&\geq \#(A)^{[1]} \cdot |\delta_{-}(C_0, x_0)| \quad (\text{nach Lemma 5}) \\
&\geq \#\{\delta_{\uparrow}(C_0, x_0 \cdot y u_1 z_y 0 0) \mid y \in \mathcal{V}(x_0)\}
\end{aligned}$$

Daraus folgt

$$\begin{aligned}
&\exists y, y' \in \mathcal{V}(x_0), a \in \Sigma_n : \\
&a \in \langle y \rangle - \langle y' \rangle, \\
&\delta_{\uparrow}(C_0, x_0 \cdot y u_1 z_y 0 0) = \delta_{\uparrow}(C_0, x_0 \cdot y' u_1 z_{y'} 0 0)
\end{aligned}$$

Definiere

$$\begin{aligned}
w &\stackrel{\text{def}}{=} \text{begin}(C_0, x_0 \cdot y u_1 z_y 0 0) 0 a \\
w' &\stackrel{\text{def}}{=} \text{begin}(C_0, x_0 \cdot y' u_1 z_{y'} 0 0) 0 a
\end{aligned}$$

Es gilt  $w \in L_n$  und  $w' \in MIP(L_n)$ .  $A$  kann  $w$  und  $w'$  jedoch nicht unterscheiden und akzeptiert somit entweder keines dieser Wörter oder beide. Dies steht im Widerspruch zur Annahme,  $A$  akzeptiere  $L_n$ .

**Fall 2:**  $\#\mathcal{U} < \#\mathcal{U}'$

Definiere für alle  $y \in \mathcal{T}_0$

$$\mathcal{V}'(y) \stackrel{\text{def}}{=} \{x \in \mathcal{T}_2 \mid (x, y) \in \mathcal{U}'\}$$

und wähle  $y_0$  so, daß  $\mathcal{V}'(y_0)$  maximale Kardinalität hat.

Es gilt

$$\begin{aligned}
\#\mathcal{V}'(y_0) &\geq \#\mathcal{U}'/\#\mathcal{T}_0 \\
&\geq \#\mathcal{T}_2/2 \\
&= 2^{n/3}/2 \\
&> |A| \\
&> \#(A)^{[1]} \\
&\geq \#\{(\delta_\uparrow(C_0, x \cdot y_0))^{[1]} \mid x \in \mathcal{V}'(y_0)\}
\end{aligned}$$

Daraus folgt

$$\begin{aligned}
&\exists x_0, x'_0 \in \mathcal{V}'(y_0), a_0 \in \Sigma_n : \\
&a_0 \in \langle x_0 \rangle - \langle x'_0 \rangle, \\
&(\delta_\uparrow(C_0, x_0 \cdot y_0))^{[1]} = (\delta_\uparrow(C_0, x'_0 \cdot y_0))^{[1]}
\end{aligned}$$

Definiere

$$\begin{aligned}
v &\stackrel{\text{def}}{=} \text{begin}(C_0, x_0 \cdot y_0) \\
v' &\stackrel{\text{def}}{=} \text{begin}(C_0, x'_0 \cdot y_0) \\
C_1 &\stackrel{\text{def}}{=} \delta_\uparrow(C_0, x_0 \cdot y_0) \\
C'_1 &\stackrel{\text{def}}{=} \delta_\uparrow(C_0, x'_0 \cdot y_0)
\end{aligned}$$

Für alle  $w \in \Sigma_{2n/3}^* 0^*$  gilt

$$\begin{aligned}
v w 0 a_0 &\in L_n, \\
v' w 0 a_0 &\in MIP(L_n)
\end{aligned}$$

Der Rest des Beweises besteht aus der rekursiven Definition eines Wortes  $w_{d+1}$ , so daß  $A$ , in der Konfiguration  $C_1$  gestartet, während der Verarbeitung von  $w_{d+1}$  insgesamt mindestens  $d+1$  Symbole auf den Keller schreibt und in keine Konfiguration übergeht, die kürzer ist als  $C_1$ . Wegen  $(C_1)^{[1]} = (C'_1)^{[1]}$  gilt dasselbe dann auch für  $C'_1$ .

**Induktionsbehauptung:**

$$\forall i \in \mathbb{N}_0 \exists w_i, v_i \in \Sigma_{2n/3}^* \forall z \in \mathcal{T}_{(i+1) \bmod 2} :$$

$$\begin{aligned} |C_1| + i &\leq |\delta_{\uparrow}(C_1, w_i \cdot v_i)| \\ &< \left| \delta_{\uparrow}(C_1, w_i v_i \cdot u_{(i+1) \bmod 2} z 0 0) \right|, \\ |C_1| &= \left| \delta_{\uparrow}(C_1, \varepsilon \cdot w_i v_i u_{(i+1) \bmod 2} z 0 0) \right| \end{aligned}$$

**Induktionsverankerung:**  $i = 0$

Definiere

$$\begin{aligned} w_0 &\stackrel{\text{def}}{=} \varepsilon \\ v_0 &\stackrel{\text{def}}{=} \text{mid-end}(C_0, x_0 \cdot y_0) \end{aligned}$$

Es gilt

$$\begin{aligned} |C_1| &= |\delta_{\uparrow}(C_0, x_0 \cdot y_0)| \\ &\leq |\delta_{\uparrow}(C_1, w_0 \cdot v_0)| \\ &= |\delta_{\uparrow}(C_1, \varepsilon \cdot w_0 v_0)| \\ &\leq |C_1| \end{aligned}$$

Wegen  $(x_0, y_0) \in \mathcal{U}'$  gilt für alle  $z \in \mathcal{T}_1$

$$\begin{aligned} |\delta_{\uparrow}(C_0, x_0 \cdot y_0)| &< |\delta_{\uparrow}(C_0, x_0 y_0 \cdot u_1 z 0 0)| \\ &= |\delta_{\uparrow}(C_1, w_0 v_0 \cdot u_1 z 0 0)| \end{aligned}$$

sowie

$$\begin{aligned} |C_1| &= \min\{|\delta_{\uparrow}(C_1, \varepsilon \cdot w_0 v_0)|, |\delta_{\uparrow}(C_1, w_0 v_0 \cdot u_1 z 0 0)|\} \\ &= |\delta_{\uparrow}(C_1, \varepsilon \cdot w_0 v_0 u_1 z 0 0)| \end{aligned}$$

Demnach erfüllen  $w_0$  und  $v_0$  die Induktionsbehauptung.

**Induktionsschritt:**  $i \rightarrow i + 1$

Die Induktionsbehauptung gelte für ein  $i \in \mathbb{N}_0$ . Definiere

$$w_{i+1} \stackrel{\text{def}}{=} w_i v_i u_{(i+1) \bmod 2}$$

### Fallunterscheidung

**Fall 1:**  $\forall y \in \mathcal{T}_{(i+1) \bmod 2} \exists z_y \in \mathcal{T}_i \bmod 2 :$

$$|\delta_{\uparrow}(C_1, w_{i+1} y \cdot u_{i \bmod 2} z_y 0 0)| \leq |\delta_{\uparrow}(C_1, w_{i+1} \cdot y)|$$

In diesem Fall ist  $\delta_{\downarrow}(C_1, w_{i+1} \cdot y \cdot u_{i \bmod 2} z_y 0 0)$  für alle  $y \in \mathcal{T}_{(i+1) \bmod 2}$  definiert, und nach Abbildung 4.1 auf Seite 51 ist der Kellerinhalt von  $\delta_{\downarrow}(C_1, w_{i+1} \cdot y \cdot u_{i \bmod 2} z_y 0 0)$  ohne das oberste Kellersymbol ein echtes Präfix des Kellerinhalts von  $\delta_{-}(C_1, w_{i+1})$ .

Mit der Induktionsvoraussetzung ergibt sich für alle  $y \in \mathcal{T}_{(i+1) \bmod 2}$

$$\begin{aligned} |\delta_{\uparrow}(C_1, w_i \cdot v_i)| &< |\delta_{\uparrow}(C_1, w_i v_i \cdot u_{(i+1) \bmod 2} y 0 0)| \\ &\leq |\delta_{\uparrow}(C_1, w_{i+1} \cdot y)| \\ &= |\delta_{\downarrow}(C_1, w_{i+1} \cdot y \cdot u_{i \bmod 2} z_y 0 0)| \\ &\leq |\delta_{-}(C_1, w_{i+1})| \\ &= |\delta_{-}(C_1, w_i v_i u_{(i+1) \bmod 2})| \end{aligned}$$

Daraus folgt

$$\begin{aligned} \#\mathcal{T}_{(i+1) \bmod 2} &= 2^{n/3} \\ &> 4 \frac{n}{3} \cdot |A|^2 \\ &\geq \#(A)^{[1]} \cdot \left( |\delta_{-}(C_1, w_i v_i u_{(i+1) \bmod 2})| - |\delta_{\uparrow}(C_1, w_i \cdot v_i)| \right) \\ &\quad (\text{nach Lemma 5}) \\ &\geq \#\left\{ \delta_{\downarrow}(C_1, w_{i+1} \cdot y \cdot u_{i \bmod 2} z_y 0 0) \mid y \in \mathcal{T}_{(i+1) \bmod 2} \right\} \end{aligned}$$

sowie

$$\exists y, y' \in \mathcal{T}_{(i+1) \bmod 2}, a \in \Sigma_n :$$

$$a \in \langle y \rangle - \langle y' \rangle,$$

$$\delta_{\downarrow}(C_1, w_{i+1} \cdot y \cdot u_{i \bmod 2} z_y 00) = \delta_{\downarrow}(C_1, w_{i+1} \cdot y' \cdot u_{i \bmod 2} z_{y'} 00)$$

Definiere

$$\begin{aligned} w &\stackrel{\text{def}}{=} v \text{ begin} - \text{mid} (C_1, w_{i+1} \cdot y \cdot u_{i \bmod 2} z_y 0 0) 0 a \\ w' &\stackrel{\text{def}}{=} v \text{ begin} - \text{mid} (C_1, w_{i+1} \cdot y' \cdot u_{i \bmod 2} z_{y'} 0 0) 0 a \end{aligned}$$

Es gilt  $w \in L_n$ , aber  $w' \in MIP(L_n)$ .  $A$  kann  $w$  und  $w'$  jedoch nicht unterscheiden und akzeptiert somit entweder keines dieser Wörter oder beide. Dies steht im Widerspruch zur Annahme,  $A$  akzeptiere  $L_n$ .

**Fall 2:**  $\exists v_{i+1} \in \mathcal{T}_{(i+1) \bmod 2} \forall z \in \mathcal{T}_{i \bmod 2} :$

$$|\delta_{\uparrow}(C_1, w_{i+1} v_{i+1} \cdot u_{i \bmod 2} z 0 0)| > |\delta_{\uparrow}(C_1, w_{i+1} \cdot v_{i+1})|$$

Mit der Induktionsvoraussetzung ergibt sich für alle  $z \in \mathcal{T}_{i \bmod 2}$

$$\begin{aligned} |C_1| + i &< |\delta_{\uparrow}(C_1, w_i v_i \cdot u_{(i+1) \bmod 2} v_{i+1} 0 0)| \\ &\leq |\delta_{\uparrow}(C_1, w_{i+1} \cdot v_{i+1})| \\ &< |\delta_{\uparrow}(C_1, w_{i+1} v_{i+1} \cdot u_{i \bmod 2} z 0 0)| \end{aligned}$$

sowie

$$\begin{aligned} |C_1| &= |\delta_{\uparrow}(C_1, \varepsilon \cdot w_i v_i u_{(i+1) \bmod 2} v_{i+1} 0 0)| \\ &= |\delta_{\uparrow}(C_1, \varepsilon \cdot w_{i+1} v_{i+1} 0 0)| \\ &\leq \min\{|\delta_{\uparrow}(C_1, \varepsilon \cdot w_{i+1} v_{i+1})|, |\delta_{\uparrow}(C_1, w_{i+1} \cdot v_{i+1})|\} \\ &\leq \min\{|\delta_{\uparrow}(C_1, \varepsilon \cdot w_{i+1} v_{i+1})|, |\delta_{\uparrow}(C_1, w_{i+1} v_{i+1} \cdot u_{i \bmod 2} z 0 0)|\} \\ &= |\delta_{\uparrow}(C_1, \varepsilon \cdot w_{i+1} v_{i+1} u_{i \bmod 2} z 0 0)| \\ &\leq |C_1| \end{aligned}$$

Demnach erfüllen  $w_{i+1}$  und  $v_{i+1}$  die Induktionsbehauptung.

**Ende des Induktionsbeweises**

Wegen  $\varepsilon \in \mathcal{T}_{(d+1) \bmod 2}$  gilt aufgrund der Induktionsaussage

$$\begin{aligned} |C_1| + d &\leq \left| \delta_{\uparrow} \left( C_1, w_d v_d \cdot u_{(d+1) \bmod 2} 0 0 \right) \right| - 1 \\ &\leq \left| \delta_{\uparrow} \left( C_1, w_d v_d \cdot u_{(d+1) \bmod 2} 0 a_0 \right) \right| \\ &\leq \left| \delta_{-} \left( C_1, w_{d+1} 0 a_0 \right) \right| \end{aligned}$$

sowie

$$\begin{aligned} |C_1| &= \left| \delta_{\uparrow} \left( C_1, \varepsilon \cdot w_d v_d u_{(d+1) \bmod 2} 0 0 \right) \right| \\ &= \left| \delta_{\uparrow} \left( C_1, \varepsilon \cdot w_{d+1} 0 0 \right) \right| \end{aligned}$$

Daraus folgt

$$\begin{aligned} (\delta_{-} (C_0, v w_{d+1} 0 a_0))^{[d]} &= (\delta_{-} (C_1, w_{d+1} 0 a_0))^{[d]} \\ &= (\delta_{-} (C'_1, w_{d+1} 0 a_0))^{[d]} \\ &= (\delta_{-} (C_0, v' w_{d+1} 0 a_0))^{[d]} \end{aligned}$$

Außerdem gilt

$$\begin{aligned} v w_{d+1} 0 a_0 &\in L_n, \\ v' w_{d+1} 0 a_0 &\in MIP(L_n) \end{aligned}$$

$A$  muß nach dem Lesen von  $v' w_{d+1} 0 a_0$  mehr als  $d$  Übergänge machen, bis er diese Eingabe von  $v w_{d+1} 0 a_0$  unterscheiden und zurückweisen kann. Dies steht im Widerspruch zur Annahme,  $A$  erkenne  $L_n$  mit Zeitverzögerung  $d$ .

Daraus folgt

$$\begin{aligned} \sigma_d^Z(L'_n) &= \sigma_d^Z \left( L_{6 \lceil \log_2 n+1 \rceil \cdot (n+5)} \right) \\ &\geq \frac{2^{\lfloor 6 \lceil \log_2 n+1 \rceil \cdot (n+5)/6 \rfloor}}{12 \lceil \log_2 n+1 \rceil \cdot (n+5)} \\ &\geq \frac{(2n)^{n+5}}{12n \cdot (n+5)} \end{aligned}$$

$$\begin{aligned} &\geq \frac{2^{n+4}}{6 \cdot (n+5)} \cdot n^n \\ &\geq n^n \end{aligned}$$

**2. Teil:** Zu zeigen:

Für alle  $n \in \mathbb{N}$  gibt es einen moderaten RDPDA  $A$  der Größe  $|A| = O(n^2)$ , der  $L_n$  mit Zeitverzögerung  $\infty$  / Eingabeverzögerung  $0$  erkennt.

Definiere

$$A \stackrel{\text{def}}{=} (Q, \Sigma'_n, \Sigma'_n, \delta, (\uparrow, 0), \{q_f\}, q_r),$$

wobei

$$Q \stackrel{\text{def}}{=} \{[\downarrow, b] \mid b \in \Sigma_n\} \cup \{\uparrow, \rightarrow, q_f, q_r\}$$

sowie für alle  $a, Z \in \Sigma'_n$  und  $b \in \Sigma_n$

$$\delta(\uparrow, Z, a) \stackrel{\text{def}}{=} \begin{cases} (\uparrow, Z a), & \text{falls } a \neq 0 \\ (\rightarrow, Z), & \text{falls } a = 0 \end{cases}$$

$$\delta(\rightarrow, Z, a) \stackrel{\text{def}}{=} \begin{cases} (\rightarrow, Z), & \text{falls } a = 0 \\ ([\downarrow, a], Z), & \text{falls } a \neq 0 \end{cases}$$

$$\delta([\downarrow, b], Z, \varepsilon) \stackrel{\text{def}}{=} \begin{cases} ([\downarrow, b], \varepsilon), & \text{falls } Z \neq b, Z \neq -b, Z \neq 0 \\ (q_f, 0), & \text{falls } Z = b \\ (q_r, \varepsilon), & \text{falls } Z = -b \vee Z = 0 \end{cases}$$

$$\delta(q_f, 0, a) \stackrel{\text{def}}{=} (q_r, \varepsilon)$$

$A$  verarbeitet eine Eingabe  $w 0^k a$  mit  $w \in \Sigma_n^*$  folgendermaßen: Im Zustand  $\uparrow$  wird  $w$  gelesen und auf den Keller geschrieben; im Zustand  $\rightarrow$  wird  $0^k a$  gelesen und schließlich nach  $[\downarrow, a]$  übergegangen. Im Zustand  $[\downarrow, a]$  leert  $A$  seinen Keller mit  $\varepsilon$ -Übergängen. Wenn er dabei das Kellersymbol  $a$  antrifft, bevor er auf  $-a$  oder das unterste Kellersymbol  $0$  stößt, dann akzeptiert er; andernfalls weist er die Eingabe zurück.

Offensichtlich erkennt  $A$  die Sprache  $L_n$  mit Eingabeverzögerung 0. Die Größe von  $A$  beträgt

$$|A| = O(n^2)$$

■

**Satz 9** *Es gibt eine Familie von (endlichen, präfixfreien) Sprachen  $(L_n)_{n \in \mathbb{N}}$  und eine Konstante  $c \geq 1$ , so daß für alle  $n \geq 2$  gilt*

$$\sigma^Z(L_n) \in \left( n^{cn} \mid n^{cn/2} \mid n^{cn/3} \mid \cdots \mid n^c \mid n^c \mid \cdots \parallel n^c \right)$$

**Beweis** Definiere für alle  $n \in \mathbb{N}$

$$\Sigma_n \stackrel{\text{def}}{=} [1 : n]$$

$$\Sigma'_n \stackrel{\text{def}}{=} [1 : n] \times [1 : n]$$

$$L_n \stackrel{\text{def}}{=} \{a_1 \cdots a_n [i, b] \mid a_1, \dots, a_n \in \Sigma_n, [i, b] \in \Sigma'_n, a_i = b\}$$

**1. Teil:** Zu zeigen:

*Für jeden RDPDA  $A$ , der  $L_n$  mit Zeitverzögerung  $d$  für ein  $n \in \mathbb{N}$  und  $d \in \mathbb{N}_{0, \infty}$  erkennt, gilt*

$$|A| \geq \max \{n, n^{n/(d+1)}\}$$

Für jeden RDPDA  $A$  mit  $L(A) = L_n$  für ein  $n \in \mathbb{N}$  gilt  $|A| \geq n^2$ , da  $A$  sonst nicht alle Eingabezeichen lesen könnte. Ein RDPDA, der  $L_n$  mit Zeitverzögerung  $d$  für ein  $d \in \mathbb{N}_0$  erkennt, muß innerhalb von  $d + 1$  Übergängen alle Wörter aus  $\Sigma_n^n$  unterscheiden können. Dies bedingt, daß es für jedes solche Wort einen eigenen fortsetzbaren  $(d + 1)$ -Modus gibt, und als untere Schranke für die Größe eines solchen RDPDA ergibt sich daraus  $n^{n/(d+1)}$ . Dies wird im folgenden formal gezeigt.

**Annahme:** Es gibt einen RDPDA  $A$  der Größe  $|A| < n^{n/(d+1)}$ , der  $L_n$  mit Zeitverzögerung  $d$  für ein  $n \in \mathbb{N}$  und  $d \in [0 : n - 1]$  erkennt.

Sei  $A = (Q, \Sigma_n \cup \Sigma'_n, \Gamma, \delta, C_0, F, q_r)$ . Wegen  $\#(A)^{[d+1]} < n^n$  gilt

$$\begin{aligned} \exists w_1, w_2 \in \Sigma_n^n, i_0 \in [1 : n], b_1 \in \Sigma_n : \\ b_1 = (w_1)_{[i_0]} \neq (w_2)_{[i_0]}, \\ (\delta_+(C_0, w_1))^{[d+1]} = (\delta_+(C_0, w_2))^{[d+1]} \end{aligned}$$

Es gilt  $w_1[i_0, b_1] \in L_n$  und  $w_2[i_0, b_1] \in MIP(L_n)$ . Da  $A$  die Sprache  $L_n$  mit Zeitverzögerung  $d$  erkennt, besteht die zurückweisende Berechnung

$$\delta_-(C_0, w_2[i_0, b_1]) \stackrel{\varepsilon}{\longleftarrow} \delta_+(C_0, w_2[i_0, b_1])$$

aus höchstens  $d$   $\varepsilon$ -Übergängen. Wegen  $(\delta_-(C_0, w_1[i_0, b_1]))^{[d]} = (\delta_-(C_0, w_2[i_0, b_1]))^{[d]}$  folgt daraus, daß  $\delta_+(C_0, w_1[i_0, b_1])$  ebenfalls zurückweisend ist. Dies steht im Widerspruch zur Annahme,  $A$  erkenne  $L_n$ .

**2. Teil:** Zu zeigen:

*Für alle  $n \in \mathbb{N}$  und  $d \in [0 : n - 1]$  gibt es einen moderaten RDPDA  $A$  der Größe  $|A| = O(n^{\lceil n/(d+1) \rceil + 3})$ , der  $L_n$  mit Zeitverzögerung  $d$  erkennt.*

Die Beweisidee ist folgende:  $A$  speichert die ersten  $n$  Zeichen einer Eingabe auf dem Keller, wobei er jeweils  $\lceil n/(d+1) \rceil$  Zeichen in einem Kellersymbol zusammenfaßt, so daß er innerhalb von  $d+1$  Übergängen auf jedes dieser Zeichen zugreifen kann. Bei der folgenden formalen Definition beachte man den Unterschied zwischen  $\varepsilon$  und  $[\varepsilon]$ .

Definiere

$$A \stackrel{\text{def}}{=} (Q, \Sigma_n \cup \Sigma'_n, \Gamma, \delta, ([\uparrow, 0], [\varepsilon]), \{q_f\}, q_r),$$

wobei

$$Q \stackrel{\text{def}}{=} \{[\uparrow, k] \mid k \in [0 : n]\} \cup \{[\downarrow, k, s] \mid k \in [0 : n], s \in \Sigma'_n\} \cup \{q_f, q_r\}$$

$$m \stackrel{\text{def}}{=} \lceil n/(d+1) \rceil$$

$$\Gamma \stackrel{\text{def}}{=} \{[x] \mid x \in \Sigma_n^{\leq m}\}$$

sowie für alle  $k \in [0 : n]$ ,  $s \in \Sigma'_n$ ,  $x \in \Sigma_n^{\leq m}$  und  $a \in \Sigma_n \cup \Sigma'_n$

$$\begin{aligned}
 \delta([\uparrow, k], [x], a) &\stackrel{\text{def}}{=} \left\{ \begin{array}{ll} ([\uparrow, k+1], [xa]), & \text{falls } k < n, |x| < m, a \in \Sigma_n \\ ([\uparrow, k+1], [x][a]), & \text{falls } k < n, |x| = m, a \in \Sigma_n \\ ([\downarrow, k-|x|, a], \varepsilon), & \text{falls } k = n, a \in \Sigma'_n, \\ & \text{first}(a) \leq k - |x| \\ (q_f, [\varepsilon]), & \text{falls } k = n, a \in \Sigma'_n, \\ & \text{first}(a) > k - |x|, \\ & x_{[\text{first}(a)-(k-|x|)]} = \text{second}(a) \\ (q_r, [\varepsilon]), & \text{falls } (k < n, a \in \Sigma'_n) \vee \\ & (k = n, a \in \Sigma_n) \vee \\ & (k = n, a \in \Sigma'_n, \\ & \text{first}(a) > k - |x|, \\ & x_{[\text{first}(a)-(k-|x|)]} \neq \text{second}(a) \end{array} \right. \\
 \delta([\downarrow, k, s], [x], \varepsilon) &\stackrel{\text{def}}{=} \left\{ \begin{array}{ll} ([\downarrow, k-m, s], \varepsilon), & \text{falls } \text{first}(s) \leq k - m \\ (q_f, [\varepsilon]), & \text{falls } \text{first}(s) > k - m, \\ & x_{[\text{first}(s)-(k-m)]} = \text{second}(s) \\ (q_r, [\varepsilon]), & \text{falls } \text{first}(s) > k - m, \\ & x_{[\text{first}(s)-(k-m)]} \neq \text{second}(s) \end{array} \right. \\
 \delta(q_f, [\varepsilon], a) &\stackrel{\text{def}}{=} (q_r, [\varepsilon])
 \end{aligned}$$

A verarbeitet eine Eingabe  $a_1 \cdots a_n [i, b]$  folgendermaßen: A liest zunächst  $a_1 \cdots a_n$  und speichert jeweils  $m$  aufeinanderfolgende Zeichen in einem Kellersymbol. Hierbei befindet sich A in Zuständen der Form  $[\uparrow, k]$ , wobei  $k$  die Anzahl der bereits gelesenen Eingabezeichen angibt. Anschließend liest A das Zeichen  $[i, b] \in \Sigma'_n$  und leert den Keller mit  $\varepsilon$ -Übergängen, um das Zeichen  $a_i$  zu finden. Hierbei befindet er sich in Zuständen der Form  $[\downarrow, k, [i, b]]$ , wobei  $k$  die Anzahl der noch auf dem Keller gespeicherten Eingabezeichen angibt. Sobald A auf  $a_i$  zugreifen kann, vergleicht er dieses Zeichen mit  $b$ .

Wegen  $\lceil n/m \rceil \leq d+1$  erkennt A die Sprache  $L_n$  mit Zeitverzögerung  $d$ . Die Größe von A beträgt

$$|A| = O\left(n^{\lceil n/(d+1) \rceil + 3}\right)$$

■

## 4.4 Das Spektrum der Gesamt-Zeitverzögerung

**Satz 10** *Es gibt eine Familie von (endlichen, präfixfreien) Sprachen  $(M'_n)_{n \in \mathbb{N}}$  und eine Konstante  $c \geq 1$ , so daß für alle  $n \geq 2$  gilt*

$$\sigma^G(M'_n) \in \left( \begin{array}{c|c|c|c|c} n^{cn} & n^{cn/2} & n^{cn/3} & \cdots & n^c \\ n^n & n^{n/2} & n^{n/3} & \cdots & n \end{array} \right)$$

**Beweis** Definiere für alle  $n \in \mathbb{N}$  die Sprache  $L_n$  wie im Beweis von Satz 9 sowie

$$\begin{aligned} M_n &\stackrel{\text{def}}{=} L_n \cdot \$ \cup MIP(L_n) \cdot \mathcal{L} \\ M'_n &\stackrel{\text{def}}{=} M_{2n} \\ &= L_{2n} \cdot \$ \cup MIP(L_{2n}) \cdot \mathcal{L} \end{aligned}$$

Dieser Beweis beruht darauf, daß es etwa gleich schwer ist,  $M_n$  mit Gesamt-Zeitverzögerung  $d$  zu erkennen, wie  $L_n$  mit Zeitverzögerung  $d$  zu erkennen.

Zum Beweis der unteren Schranke sei ein RDPDA

$$A = (Q, \Sigma_n \cup \Sigma'_n \cup \{\$, \mathcal{L}\}, \Gamma, \delta, (q_0, Z_0), F, q_r)$$

gegeben, der  $M_n$  mit Gesamt-Zeitverzögerung  $d$  für ein  $d \in \mathbb{N}_{0,\infty}$  erkennt. Es wird ein RDPDA  $A'$  konstruiert, der  $L_n$  mit Zeitverzögerung  $d$  erkennt.  $A'$  simuliert im wesentlichen  $A$  und kann durch Speichern des obersten Kellersymbols von  $A$  in der endlichen Kontrolle außerdem frühzeitig vorhersehen, ob  $A$  eine Eingabe aus  $(L_n \cup MIP(L_n)) \cdot \$$  akzeptiert oder zurückweist.

Definiere

$$A' \stackrel{\text{def}}{=} (Q', \Sigma_n \cup \Sigma'_n, \Gamma', \delta', ([q_0, Z_0], Z'_0), \{q_f\}, q_r),$$

wobei

$$Q' \stackrel{\text{def}}{=} ((Q - \{q_r\}) \times \Gamma) \cup \{q_f, q_r\}$$

$$\Gamma' \stackrel{\text{def}}{=} \Gamma \cup \{Z'_0\}$$

$\delta'$  wird durch folgende Regeln definiert:

1. Falls  $\delta(q, Z, a) = (p, \gamma)$  mit  $p \in Q$  und  $\gamma \in \Gamma^*$  für ein  $q \in Q$ ,  $Z \in \Gamma$  und  $a \in \Sigma_n \cup \Sigma'_n \cup \{\varepsilon\}$ , dann sei für alle  $Z' \in \Gamma$

$$\delta'([q, Z], Z', a) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} \left( [p, (Z' \gamma)^{[1]}], (Z' \gamma)^{-[1]} \right), \\ \text{falls } (p, Z' \gamma) \text{ eine } \varepsilon\text{-Konfiguration von } A \text{ ist } \vee \\ \left( (p, Z' \gamma) \text{ eine Lese-Konfiguration von } A \text{ ist,} \right. \\ \delta(p, Z' \gamma, \$) \in \{q_r\} \times \Gamma^*, \\ \left. \delta(p, Z' \gamma, \mathcal{L}) \in \{q_r\} \times \Gamma^* \right) \\ (q_f, Z'_0), \\ \text{falls } (p, Z' \gamma) \text{ eine Lese-Konfiguration von } A \text{ ist,} \\ \delta(p, Z' \gamma, \$) \notin \{q_r\} \times \Gamma^*, \\ \delta(p, Z' \gamma, \mathcal{L}) \in \{q_r\} \times \Gamma^* \\ (q_r, \varepsilon), \\ \text{falls } (p, Z' \gamma) \text{ eine Lese-Konfiguration von } A \text{ ist,} \\ \delta(p, Z' \gamma, \$) \in \{q_r\} \times \Gamma^*, \\ \delta(p, Z' \gamma, \mathcal{L}) \notin \{q_r\} \times \Gamma^* \end{array} \right.$$

2. Für alle  $a \in \Sigma_n \cup \Sigma'_n$  sei

$$\delta'(q_f, Z'_0, a) \stackrel{\text{def}}{=} (q_r, \varepsilon)$$

Da  $A$  die Sprache  $M_n$  mit Zeitverzögerung 0 erkennt, gilt für alle  $x \in (\Sigma_n \cup \Sigma'_n)^*$

$$\begin{aligned} x \in L_n &\iff \delta_-((q_0, Z_0), x \$) \notin \{q_r\} \times \Gamma^*, \\ x \in MIP(L_n) &\iff \delta_-((q_0, Z_0), x \mathcal{L}) \notin \{q_r\} \times \Gamma^* \end{aligned}$$

$A'$  simuliert  $A$  so lange, bis  $A$  einen Übergang  $C \xrightarrow{\frac{a}{A}} C'$  ausführt, wobei  $C'$  eine Lese-Konfiguration und entweder  $\delta(C', \$)$  oder  $\delta(C', \mathcal{L})$  keine zurückweisende Konfiguration ist. Sei  $x$  die bis zur Konfiguration  $C'$  gelesene Eingabe. Falls  $\delta(C', \$)$  nicht zurückweisend ist, dann gilt  $x \in L_n$ , und  $A'$  geht in den akzeptierenden Zustand  $q_f$  über; andernfalls gilt

$x \in MIP(L_n)$ , und  $A'$  geht in den zurückweisenden Zustand  $q_r$  über. Da  $A$  auf keiner Eingabe mehr als  $d$   $\varepsilon$ -Übergänge macht, erkennt  $A'$  die Sprache  $M_n$  mit Zeitverzögerung  $d$ . Wegen  $|A'| \leq |A|^2$  folgt die untere Schranke mit der Definition von  $M'_n$  aus Satz 9.

Zum Beweis der oberen Schranke definiere gemäß Satz 9 einen moderaten RDPDA

$$A = (Q, \Sigma_n \cup \Sigma'_n, \Gamma, \delta, (q_0, Z_0), \{q_f\}, q_r),$$

der  $L_n$  mit Zeitverzögerung  $d$  für ein  $d \in [0 : n - 1]$  erkennt, und dessen einziger erreichbarer, zurückweisender Modus  $(q_r, Z_0)$  ist.  $A$  kann leicht zu einem moderaten RDPDA  $A'$  erweitert werden, der  $M_n$  mit Gesamt-Zeitverzögerung  $d$  erkennt.

Definiere

$$A' \stackrel{\text{def}}{=} (Q \cup \{q'_f, q'_r\}, \Sigma_n \cup \Sigma'_n \cup \{\$, \mathcal{L}\}, \Gamma, \delta \cup \delta', (q_0, Z_0), \{q'_f\}, q'_r)$$

$\delta'$  wird durch folgende Regeln definiert:

1. Falls  $(q, Z) \in Q \times \Gamma$  ein erreichbarer Lese-Modus oder Stop-Modus von  $A$  ist, dann sei

$$\delta'(q, Z, \$) \stackrel{\text{def}}{=} \begin{cases} (q'_f, Z_0), & \text{falls } q = q_f \\ (q'_r, \varepsilon), & \text{falls } q \neq q_f \end{cases}$$

$$\delta'(q, Z, \mathcal{L}) \stackrel{\text{def}}{=} \begin{cases} (q'_f, Z_0), & \text{falls } q = q_r \\ (q'_r, \varepsilon), & \text{falls } q \neq q_r \end{cases}$$

2. Für alle  $a \in \Sigma_n \cup \Sigma'_n$  sei

$$\delta'(q_r, Z_0, a) \stackrel{\text{def}}{=} (q'_r, \varepsilon)$$

3. Für alle  $a \in \Sigma_n \cup \Sigma'_n \cup \{\$, \mathcal{L}\}$  sei

$$\delta'(q'_f, Z_0, a) \stackrel{\text{def}}{=} (q'_r, \varepsilon)$$

Da  $A$  auf ableitbaren Konfigurationen keinen  $\varepsilon$ -Übergang ausführt, erkennt  $A'$  die Sprache  $M_n$  mit Gesamt-Zeitverzögerung  $d$ . Wegen  $|A'| = O(|A|)$  folgt die obere Schranke mit der Definition von  $M'_n$  aus Satz 9. ■

**Satz 11** *Es gibt eine Familie von (regulären) Sprachen  $(N'_n)_{n \in \mathbb{N}}$  und eine Konstante  $c \geq 1$ , so daß für alle  $n \geq 2$  gilt*

$$\sigma^G(N'_n) \in \left( \begin{array}{c} | \\ n^n \\ | \\ n^n \\ | \\ \cdots \\ | \\ n^n \\ | \\ \cdots \\ | \\ n^c \end{array} \right)$$

**Beweis** Definiere für alle  $n \in \mathbb{N}$  die Sprachen  $M_n$  und  $M'_n$  wie im Beweis von Satz 10 sowie

$$\begin{aligned} N_n &\stackrel{\text{def}}{=} (M_n)^* \\ N'_n &\stackrel{\text{def}}{=} N_{4n} \\ &= (M'_{2n})^* \end{aligned}$$

Dieser Beweis beruht darauf, daß es etwa gleich schwer ist,  $(M_n)^*$  mit endlicher Gesamt-Zeitverzögerung zu erkennen, wie  $M_n$  mit Gesamt-Zeitverzögerung 0 zu erkennen.

Zum Beweis der unteren Schranke sei ein RDPDA

$$A = (Q, \Sigma_n \cup \Sigma'_n \cup \{\$, \mathcal{L}\}, \Gamma, \delta, (q_0, Z_0), F, q_r)$$

gegeben, der  $(M_n)^*$  mit endlicher Gesamt-Zeitverzögerung erkennt. Es gibt ein Wort  $w \in (M_n)^*$ , während dessen Verarbeitung  $A$  eine maximale Anzahl von  $\varepsilon$ -Übergängen ausführt, so daß der Automat  $A$  mit Startkonfiguration  $\delta_+((q_0, Z_0), w)$  die Sprache  $(M_n)^*$  mit Gesamt-Zeitverzögerung 0 erkennt. Wenn  $|\delta_+((q_0, Z_0), w)| > 1$  gilt, dann ist dieser Automat zwar kein RDPDA im strengen Sinne, es ist jedoch leicht möglich, mit derselben Idee einen RDPDA  $A'$  zu konstruieren, der  $M_n$  mit Gesamt-Zeitverzögerung 0 erkennt.

Definiere

$$A' \stackrel{\text{def}}{=} ((Q - F) \cup \{q'_0, q_f\}, \Sigma_n \cup \Sigma'_n \cup \{\$, \mathcal{L}\}, \Gamma, \delta', (q'_0, Z_0), \{q_f\}, q_r)$$

$\delta'$  wird durch folgende Regeln definiert:

1. Für alle  $a \in \Sigma_n \cup \Sigma'_n \cup \{\$, \mathcal{L}\}$  sei

$$\delta'(q'_0, Z_0, a) \stackrel{\text{def}}{=} (\delta_+((q_0, Z_0), wa))^{[n+2]}$$

2. Falls  $\delta(q, Z, a) = (p, \gamma)$  mit  $p \in Q$  und  $\gamma \in \Gamma^*$  für ein  $q \in Q - F$ ,  $Z \in \Gamma$  und  $a \in \Sigma_n \cup \Sigma'_n \cup \{\$, \mathcal{L}\}$ , dann sei

$$\delta'(q, Z, a) \stackrel{\text{def}}{=} \begin{cases} (p, \gamma), & \text{falls } p \notin F \\ (q_f, Z_0), & \text{falls } p \in F \end{cases}$$

3. Für alle  $a \in \Sigma_n \cup \Sigma'_n \cup \{\$, \mathcal{L}\}$  sei

$$\delta'(q_f, Z_0, a) \stackrel{\text{def}}{=} (q_r, \varepsilon)$$

Von der nichtakzeptierenden Startkonfiguration  $(q'_0, Z_0)$  aus geht  $A'$  beim Lesen des ersten Zeichens  $a$  nach Regel 1 in den  $(n+2)$ -Modus derjenigen Konfiguration über, die  $A$  nach dem Lesen von  $wa$  erreicht. Da keine originäre Berechnung von  $A'$  aus mehr als  $n+3$  Übergängen bestehen kann, ist dies ausreichend. Anschließend simuliert  $A'$  den RDPDA  $A$  nach Regel 2 so lange, bis  $A$  erstmals einen akzeptierenden Zustand erreicht.  $A'$  erkennt  $M_n$  mit Gesamt-Zeitverzögerung 0. Wegen  $|A'| \leq (n+4) \cdot |A| \leq |A|^2$  folgt die untere Schranke mit der Definition von  $N'_n$  aus Satz 10.

Zum Beweis der oberen Schranke definiere gemäß Satz 10 einen moderaten RDPDA

$$A = (Q, \Sigma_n \cup \Sigma'_n \cup \{\$, \mathcal{L}\}, \Gamma, \delta, (q_0, Z_0), \{q_f\}, q_r),$$

der  $M_n$  mit Gesamt-Zeitverzögerung  $n-1$  erkennt. Durch „Umleiten“ aller Übergänge von  $A$ , die in den bisherigen akzeptierenden Zustand  $q_f$  führen, zum Startzustand  $q_0$ , der einziger akzeptierender Zustand wird, erhält man einen moderaten RDPDA  $A'$  mit  $|A'| \leq |A|$ , der  $(M_n)^*$  mit Gesamt-Zeitverzögerung  $\infty$  erkennt. Die obere Schranke folgt nun mit der Definition von  $N'_n$  aus Satz 10. ■

# Kapitel 5

## Schlußbemerkung

In dieser Arbeit wurde die Abhängigkeit der Größe eines zurückweisenden DPDA von der Verzögerung, mit der dieser fehlerhafte Eingaben zurückweist, untersucht. Um diese Abhängigkeit quantitativ beschreiben zu können, wurden zunächst drei Verzögerungsmaße für zurückweisende DPDAs definiert, und darauf basierend wurden für jede DCFL  $L$  drei Verzögerungsspektren definiert. Diese Zahlenfolgen geben an, wie schwer es einem zurückweisenden DPDA fällt,  $L$  zu akzeptieren und minimale inkorrekte Präfixe von  $L$  früh zurückzuweisen.

Es konnten notwendige und hinreichende Bedingungen dafür gezeigt werden, daß eine Zahlenfolge näherungsweise das Spektrum der Eingabeverzögerung einer DCFL ist. Damit sind diejenigen Fragen, die den Einfluß der Eingabeverzögerung auf die Größe von Parsern betreffen, im wesentlichen geklärt. Für den Größenunterschied in Spektren der Zeitverzögerung konnten scharfe obere Schranken gezeigt werden; für die Spektren der Gesamt-Zeitverzögerung wurde hingegen gezeigt, daß der Beweis für die Existenz einer solchen oberen Schranke mindestens so schwer ist wie derjenige für die Entscheidbarkeit des bislang ungelösten Teilklassenproblems  $SUBCLASS(\mathcal{D}, \mathcal{R})$ .

Manche interessanten Fragen konnten in dieser Arbeit nicht beantwortet werden. Zuerst ist hier das Problem der effektiven Berechenbarkeit von Verzögerungsspektren zu nennen, wobei weiterführende Ergebnisse hierzu von Lösungen zu  $EQUIV(D, R)$  und / oder  $SUBCLASS(\mathcal{D}, \mathcal{R})$  abhängen. Wichtiger noch wäre die Kenntnis von notwendigen und hinreichenden Bedingungen dafür, daß eine Zahlenfolge (näherungsweise) das Spektrum

der Zeitverzögerung bzw. der Gesamt-Zeitverzögerung einer DCFL ist, vergleichbar denjenigen für das Spektrum der Eingabeverzögerung.

Eine vielversprechende Erweiterung des Verzögerungskonzepts besteht in der genaueren Untersuchung derjenigen Situationen, die in dieser Arbeit mit Eingabeverzögerung  $\infty$ , Zeitverzögerung  $\infty$  bzw. Gesamt-Zeitverzögerung  $\infty$  bezeichnet wurden. So wäre es insbesondere möglich, die Verzögerung eines Automaten, also die Anzahl von Übergängen eines bestimmten Typs, nicht durch eine Konstante zu beschränken, sondern durch eine (lineare) Funktion in Abhängigkeit von der Eingabelänge. Ein DPDA mit *multiplikativer* Eingabeverzögerung  $d$  dürfte z. B. nach dem Lesen eines minimalen inkorrekten Präfixes der Länge  $n$  höchstens noch  $dn$  weitere Zeichen lesen.

# Literaturverzeichnis

- [1] A. V. AHO AND J. D. ULLMAN, *Optimization of LR(k) parsers*, J. Comput. System Sci., 6 (1972), pp. 573–602.
- [2] —, *The Theory of Parsing, Translation and Compiling*, vol. I: Parsing, Prentice Hall, 1972.
- [3] —, *Error detection in precedence parsers*, Math. Syst. Theory, 7 (1973), pp. 97–113.
- [4] —, *The Theory of Parsing, Translation and Compiling*, vol. II: Compiling, Prentice Hall, 1973.
- [5] B. COURCELLE, *A representation of trees by languages I*, Theoretical Comput. Sci., 6 (1978), pp. 255–279.
- [6] J. EARLEY, *An Efficient Context-Free Parsing Algorithm*, PhD thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa., 1968.
- [7] M. M. GELLER, S. L. GRAHAM, AND M. A. HARRISON, *Production prefix parsing (extended abstract)*, in Automata, Languages and Programming, Second Colloquium, J. Loeckx, ed., vol. 14 of Lecture Notes in Computer Science, University of Saarbrücken, Springer-Verlag, August 1974, pp. 232–241.
- [8] M. M. GELLER, H. B. HUNT III, T. G. SZYMANSKI, AND J. D. ULLMAN, *Economy of description by parsers, DPDA's and PDA's*, Theoretical Comput. Sci., 4 (1977), pp. 143–153.
- [9] J. GOLDSTINE, *Formal languages and their relation to automata: What Hopcroft & Ullman didn't tell us*, in Formal Language Theory, R. V. Book, ed., Academic Press, 1980, pp. 109–140.

- [10] J. GOLDSTINE, C. KINTALA, AND D. WOTSCHKE, *On measuring nondeterminism in regular languages*, Information and Computation, 86 (1990), pp. 179–194.
- [11] M. A. HARRISON, *Introduction to Formal Language Theory*, Addison-Wesley Publishing Co., 1978.
- [12] M. A. HARRISON AND I. M. HAVEL, *Real-time strict deterministic languages*, SIAM J. Comput., 1 (1972), pp. 333–349.
- [13] ———, *On the parsing of deterministic languages*, J. Assoc. Comput. Mach., 21 (1974), pp. 525–548.
- [14] J. E. HOPCROFT AND J. D. ULLMAN, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Co., 1979.
- [15] D. E. KNUTH, *On the translation of languages from left to right*, Information and Control, 8 (1965), pp. 607–639.
- [16] H. LEUNG, *Lower bounds on size of parsers*. Unveröffentlichtes Manuskript, 1991.
- [17] A. V. MOURA, *On the cost of the viable-prefix property in precedence parsers*, Technical Report CCB 029, IBM Scientific Center, Brasilia, Brazil, 1985.
- [18] ———, *Early error detection in syntax driven parsers*, IBM J. Res. Dev., 30 (1986), pp. 617–626.
- [19] M. OYAMAGUCHI, *The equivalence problem for real-time DPDAs*, Comm. ACM, 34 (1987), pp. 731–760.
- [20] M. OYAMAGUCHI, Y. INAGAKI, AND N. HONDA, *A real-time strictness test for deterministic pushdown automata*, Information and Control, 47 (1980), pp. 1–9.
- [21] A. L. ROSENBERG, *Real-time definable languages*, J. Assoc. Comput. Mach., 14 (1967), pp. 645–662.
- [22] R. E. STEARNS, *A regularity test for pushdown machines*, Information and Control, 11 (1967), pp. 323–340.
- [23] L. G. VALIANT, *Regularity and related problems for deterministic pushdown machines*, J. Assoc. Comput. Mach., 22 (1975), pp. 1–10.