Beschreibungskomplexität von CD Grammatiksystemen

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Informatik und Mathematik
der Johann Wolfgang Goethe–Universität
in Frankfurt am Main

von

Bettina Sunckel

aus Frankfurt am Main

Frankfurt am Main 2007
(D 30)

vom Fachbereich Informatik und Mathematik der
Johann Wolfgang Goethe-Universität als Dissertation angenommen.

Dekan:	Prof. Dr. Detlef Krömker
Gutachter:	Prof. Dr. Detlef Wotschke
	Prof. Dr. Jürgen Dassow
Datum der Disputation:	24. Mai 2007

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Als formale Sprache bezeichnet man eine Menge von Wörtern, wobei ein Wort eine endliche Zeichenkette aus Symbolen eines endlichen Alphabets ist. In der Theorie der formalen Sprachen stehen mehrere Möglichkeiten zur Beschreibung einer solchen formalen Sprache zur Verfügung. Die Beschreibungen unterscheiden sich zwar durch ihre Form, sie sind jedoch alle endlich. Mit diesen endlichen Beschreibungen kann man dann sowohl endliche als auch unendliche Sprachen darstellen. Die zwei wichtigsten Klassen von Beschreibungen für formale Sprachen sind Automaten und Grammatiken.

Formale Sprachen finden heute in vielen Gebieten ihre Anwendung. Beispielsweise benutzt man sie zur Beschreibung von Programmiersprachen. Programmiersprachen dienen der Erstellung von Arbeitsanweisungen an Rechnersysteme und können als formale Sprachen gesehen werden.

Das eben angegebene Beispiel ist in der praktischen Informatik anzusiedeln, die Anwendung von formalen Sprachen im Bereich der theoretischen Informatik geht allerdings viel weiter. So hat man mit dem Modell der Turingmaschine eine formale Darstellung für alle Probleme, die durch einen Algorithmus lösbar sind. Die Turingmaschine ist ein Automatenmodell, das mit Hilfe der drei Operationen Lesen, Schreiben und Bewegen auf einem unendlichen Speicherband mit diskreten Speicherzellen und einer endlichen Menge von Zuständen Berechnungen ausführt. Eine Turingmaschine akzeptiert eine formale Sprache, wenn sie ausschließlich bei Eingabe eines Wortes aus dieser Sprache nach endlich vielen Schritten in einen der definierten Endzustände übergeht. Mit dem Modell der Turingmaschine können Probleme klassifiziert werden, Aussagen über Lösbarkeit von Problemen gemacht werden und auch der Verbrauch von Ressourcen wie Speicherplatz und Zeit bei der Lösung eines Problems untersucht werden.

Obwohl es sehr unwahrscheinlich ist, daß man natürliche Sprachen jemals vollständig durch formale Sprachen beschreiben können wird, ist es möglich, Erkenntnisse der formalen Sprachen auf Teilbereiche der natürlichen Sprachen anzuwenden. Dies ist ein Gebiet der Linguistik. Darüber hinaus finden formale Sprachen eine Anwendung in der Biologie zur Beschreibung von Mechanismen wie zum Beispiel dem Pflanzenwachstum oder anderen Arten von Zellteilung.

Ein wichtiges Ergebnis in der Theorie der formalen Sprachen ist die Chomskyhierarchie [HU79]. Sie besteht aus vier grundlegenden Klassen von formalen Sprachen, die echt ineinander enthalten sind. Diese Klassen finden in vielen Gebieten ihre Anwendung. Im einzelnen sind das die regulären Sprachen, die kontextfreien Sprachen, die kontextsensitiven Sprachen und die rekursiv aufzählbaren Sprachen. Für jede dieser Klassen gibt es ein Grammatikmodell, das die jeweilige Sprachklasse generiert, und ein Automatenmodell, das die jeweilige Sprachklasse akzeptiert.

Alle Klassen von Grammatiken, von denen Sprachklassen aus der Chomskyhierarchie erzeugt werden, haben gemeinsam, daß in den zugehörigen Grammatikmodellen von einem Startsymbol ausgehend durch sequentielles Anwenden bestimmter Ersetzungsregeln Wörter erzeugt werden können. Diese Ersetzungsregeln oder auch Produktionen sind in einer Regelmenge zusammengefaßt. Die verschiedenen Modelle unterscheiden sich nur durch die Form der Regeln. Dabei bestehen die Produktionen aus zwei Sorten von Symbolen, den Terminalen, aus denen auch die späteren Wörter bestehen, und den Nichtterminalen, die nur als Hilfssymbole zur Erzeugung von Wörtern benutzt werden. Die Zeichenkette aus Terminalen und Nichtterminalen, auf die während des Ersetzungsvorgangs Produktionen angewendet werden, heißt Satzform.

Als intuitiv besonders einfach verständliche Art von Ersetzungsregeln haben sich die kontextfreien Produktionen erwiesen. Mit der Klasse der kontextfreien Grammatiken, die aus einer Menge von Hilfssymbolen, einer Menge von Alphabetssymbolen und einer Menge von kontextfreien Produktionen bestehen, erzeugt man gerade die Klasse der kontextfreien Sprachen. Eine kontextfreie Ersetzungsregel zeichnet sich dadurch aus, daß genau ein Hilfssymbol durch eine Zeichenkette aus Hilfssymbolen und Alphabetssymbolen ersetzt wird. Die Ersetzung ist also unabhängig von der Umgebung oder dem Kontext, in dem sich das Hilfssymbol befindet. In vielen Situationen braucht man aber eine stärkere Klasse als die kontextfreien Sprachen, um Phänomene aus verschiedensten Bereichen darzustellen. In [DP89] sind einige Beispiele unter anderem aus den Gebieten der natürlichen Sprachen, der Programmiersprachen, der Logik und der Biologie aufgeführt. Die in der Chomskyhierarchie über den kontextfreien Sprachen stehende Klasse der kontextsensitiven Sprachen wird allerdings durch Ersetzungsregeln erzeugt, deren Funktionsweise viel schwerer zu erfassen ist.

Deshalb gab es mehrere Ansätze, Mechanismen zu finden, die nicht-kontextfreie Sprachen erzeugen können, aber die Einfachheit von kontextfreien Produktionen beibehalten. Viele neuere Modelle in der Theorie der formalen Sprachen erfüllen diese Anforderung.

Zum Beispiel laufen natürliche Prozesse oft nicht sequentiell sondern parallel ab. Es ist also naheliegend, einen Ersetzungsprozeß zu betrachten, der nicht jeweils eine Ersetzungsregel pro Schritt anwendet, sondern bei dem alle Symbole parallel durch jeweils eine Produktion ersetzt werden. Ein solches Ersetzungssystem wird L-System [RS80] genannt. Das Modell hat einen biologischen Hintergrund und wurde von Aristid Lindenmayer 1968 im Zusammen-

hang mit der Modellierung von Organismen, die aus mehreren Zellen bestehen, definiert. Speziell seien hier die ET0L-Systeme genannt, die aus mehreren Mengen von kontextfreien Ersetzungsregeln bestehen. Bei einem Ersetzungsschritt dürfen nur Regeln aus einer Menge benutzt werden. Mit Hilfe von kontextfreien Produktionen können im Fall der ET0L-Systeme auch nicht-kontextfreie Sprachen erzeugt werden. Die ET0L-Sprachen enthalten alle kontextfreien Sprachen und erzeugen eine echte Untermenge der kontextsensitiven Sprachen.

Eine weitere Methode zur Erweiterung kontextfreier Grammatiken ist das sogenannte „regulated rewriting“ [DP89]. Mit ihrer Hilfe wurden Matrixgrammatiken, programmierte Grammatiken, random context Grammatiken und einige weitere neuere Modelle in der Theorie der formalen Sprachen definiert. Alle haben gemeinsam, daß eine Menge oder mehrere Mengen von kontextfreien Produktionen mit Mechanismen zur Kontrolle der Anwendung dieser Produktionen versehen wird. So können auch nicht-kontextfreie Sprachen erzeugt werden.

Wenn man jetzt im Gegensatz zum „regulated rewriting“ eine kontextfreie Grammatik nicht mit mit Kontrollmechanismen ergänzt, sondern eine Menge von kontextfreie Grammatiken zusammenarbeiten läßt, so erhält man ein Grammatiksystem. Das Modell entstand durch die Übertragung sogenannter Multiagentensysteme auf Grammatiken. Bei einem Multiagentensystem stehen zur Lösung eines bestimmten Problems nur unabhängige Einheiten (auch Agenten genannt) mit eingeschränkten Fähigkeiten zur Verfügung. Ein einzelner Agent kann das gegebene Problem nicht lösen. In solch einer Situation kann man probieren, das Problem mit mehreren Agenten, die auf eine genau definierte Weise miteinander kommunizieren können, zu lösen.

Dieses Konzept der Problemlösung mittels verteilter Ressourcen wurde im „blackboard model of problem solving“ aus dem Gebiet der künstlichen Intelligenz angewendet. Das „blackboard model“ ist eine Strategie zur Lösung intellektueller Aufgaben in einer dezentralisierten Umgebung bestehend aus kommunizierenden Einheiten. Komponenten dieser Modelle sind mehrere unabhängige Wissensquellen (auch Agenten genannt) und eine Kontrolleinheit, die bestimmt, welche Wissensquelle wann und in welcher Weise zur Lösung der Aufgabe beiträgt. Es gibt ein paralleles Modell, bei dem mehrere Agenten parallel an Teilen der Lösung eines Problems arbeiten und dann ihre Ergebnisse kombinieren und ein sequentielles Modell, bei dem zu jedem Zeitpunkt nur ein Agent an dem gemeinsamen Problem arbeitet. Wenn er fertig ist, kommt ein weiterer Agent an die Reihe. Alle Details des Modells und der Zusammenhang mit Grammatiksystemen finden sich in [CVDKP94].

Wenn wir jetzt die Problemlösungsstrategie des „blackboard model of problem solving“ auf Grammatiken übertragen, erhalten wir eine Menge einzelner Grammatiken, auch Grammatiksystem genannt. Die einzelnen Grammatiken entsprechen dabei den Agenten. Mehrere Grammatiken arbeiten zusammen, um ein Wort zu erzeugen. Es gibt parallele Grammatiksysteme, bei denen zu einem Zeitpunkt mehrere Grammatiken an einer Satzform arbeiten und se-

quentielle Grammatiksysteme, bei denen zu jedem Zeitpunkt nur eine Grammatik an der gemeinsamen Satzform arbeitet. Beide Typen werden ausführlich in [CVDKP94] besprochen.

Die Struktur der sequentiellen Grammatiksysteme ist dadurch, daß die Kooperation der Komponenten nicht durch zusätzliche Strukturen oder Symbole koordiniert werden muß, sehr einfach. Trotz dieser Einfachheit wird eine stark erweiterte generative Mächtigkeit bei kontextfreien Komponenten erreicht. Daher bieten sich sequentielle oder auch „cooperating distributed“ (CD) Grammatiksysteme für die weitere Untersuchung an. Wir werden uns speziell mit kontextfreien CD Grammatiksystemen beschäftigen, weil sie auch die vorher gestellte Anforderung, mit kontextfreien Produktionen nicht-kontextfreie Sprachen erzeugen zu können, erfüllen.

Kontextfreie CD Grammatiksysteme sind eine Generalisierung von kontextfreien Grammatiken. Bei kontextfreien Grammatiken wird eine einzige Menge von kontextfreien Ersetzungsregeln benutzt, um von einem Startsymbol aus ein Wort zu erzeugen. Eine beispielhafte Darstellung dieser Vorgehensweise findet sich in Abbildung 1.1.

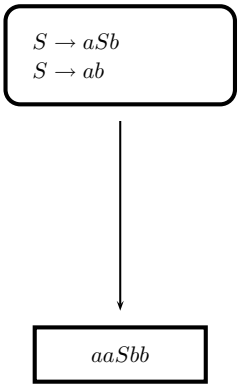

Abbildung 1.1: Grammatik - eine Menge von Produktionen arbeitet an einer Satzform.

Im Gegensatz dazu benutzt man bei CD Grammatiksystemen mehrere Regelmengen, die auch Komponenten genannt werden, und auf unterschiedliche Art und Weise miteinander kooperieren können. Zu einem Zeitpunkt arbeitet nur eine Komponente an der Satzform. Eine beispielhafte Darstellung dieser Vorgehensweise findet sich in Abbildung 1.2.

Der Ableitungsmodus bestimmt die Art und Weise, in der die Komponenten miteinander kooperieren. Die Komponente, die gerade Produktionen anwendet, nennt man auch die aktive Komponente. Zu jedem Zeitpunkt ist immer nur eine Komponente aktiv. Wir untersuchen den $*$-Modus, bei dem jede Komponente beliebig viele Produktionen anwenden darf, bevor die nächste aktiv wird und den t-Modus, bei dem jede Komponente so lange Produktionen anwenden muß, bis sie kein Nichtterminal mehr ersetzen kann. Weiterhin beschäftigen wir uns mit den Modi $(=k)$, $(\geq k)$ und $(\leq k)$, bei denen eine Kom-

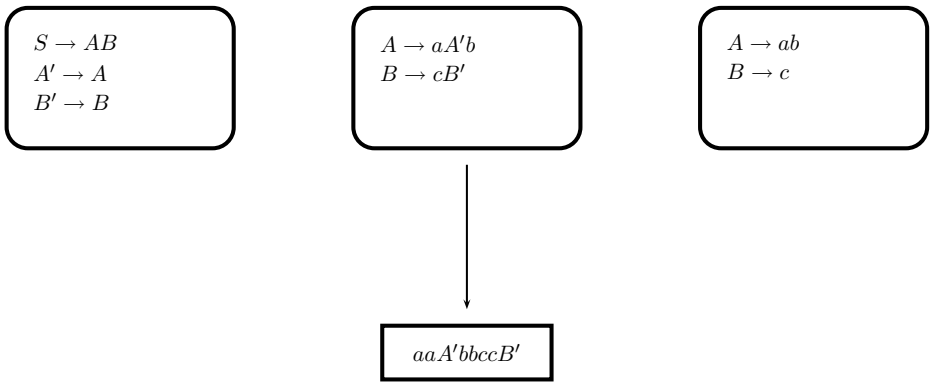

Abbildung 1.2: CD Grammatiksystem - mehrere Komponenten arbeiten an einer Satzform.

ponente jeweils genau k, mehr als k beziehungsweise weniger als k Produktionen anwenden muß, bevor die nächste Komponente aktiv wird.

CD Grammatiksysteme wurden in [MR78] definiert und eingeführt. Eine Verbindung mit Themen der künstlichen Intelligenz wurde in [CVD90] gezogen. In dieser Arbeit begann auch die ausführliche Erforschung des Modells mit der Einführung der klassischen Ableitungsmodi t, $*$, $(= k)$ und $(\leq k)$. Die $(\geq k)$-Ableitungsmodi stammen aus [DP90].

Alle CD Grammatiksysteme haben gemeinsam, daß die Komponenten im gleichen Ableitungsmodus arbeiten. Das heißt, daß der Ableitungsmodus dem ganzen Grammatiksystem zugeordnet wird und nicht den einzelnen Komponenten. Besonders im Hinblick auf das „blackboard model of problem solving“ ist diese Beschränkung aber unnatürlich. Warum sollten verschiedene Quellen mit unterschiedlichem Wissen nicht auch auf unterschiedliche Art und Weise arbeiten? Diese Frage führt uns zu hybriden CD Grammatiksystemen.

Bei hybriden CD Grammatiksystemen hat jede Komponente ihren eigenen Ableitungsmodus. Dieses Modell wurde in [Mit93] eingeführt und eingehend untersucht. Neben den oben erklärten Ableitungsmodi bietet sich hier noch ein weiterer an, der mit dem Begriff „fairness“ im Zusammenhang steht. Ein kooperierendes System wird als fair bezeichnet, wenn alle Komponenten einen ähnlich großen Beitrag zur gesamten Arbeit des Systems leisten [DM96]. Eine Komponente eines hybriden CD Grammatiksystem arbeitet im $\overline{t}$-Modus, wenn sie nicht alle Ersetzungsschritte ausführt, die möglich sind, sondern vorher eine andere Komponente aktiv wird. Da mit diesem Modus keine Terminalwörter erreicht werden können (die letzte aktive Komponente ersetzt alle in der Satzform vorhandenen Nichtterminale), wird er nur bei hybriden CD Grammatiksystemen benutzt. Der $\overline{t}$-Modus wird in [BH00] genau untersucht.

Zusammenfassend läßt sich sagen, daß mit den CD Grammatiksystemen ein vielfältiges und vielseitiges Modell gefunden wurde, indem Problemlösungsstrategien innerhalb verteilter Systeme auf Grammatiken übertragen wurden.

Mit den CD Grammatiksystemen hat man darüber hinaus ein weiteres Modell, mit dem man nicht-kontextfreie Sprachen mit Hilfe von kontextfreien Produktionen erzeugen kann. Trotzdem ist der Formalismus noch sehr einfach und intuitiv gut verständlich. Wir werden in dieser Arbeit nicht nur die eben vorgestellten uneingeschränkten Modelle untersuchen, sondern auch Einschränkungen finden, die die positiven Eigenschaften der CD Grammatiksysteme nicht wesentlich beeinflussen, aber die Struktur der Systeme und damit auch deren Handhabung stark vereinfachen.

In der Theorie der formalen Sprachen wird klassischerweise die generative Mächtigkeit eines Modells betrachtet. Es ist allerdings nicht nur wichtig, welche Sprachklasse von einem Modell beschrieben wird, sondern auch wie gut ein Modell diese Sprachklasse beschreibt. Will man jetzt genau definieren, was unter einer guten Beschreibung zu verstehen ist, liegt es nahe, daß sie so knapp oder kurz wie möglich sein sollte. Außerdem kann man von einer guten Beschreibung fordern, daß sie möglichst einfach zu verstehen sein sollte. Diese Größe und Komplexität eines Modells wird in der Beschreibungskomplexität genauer untersucht.

Die zwei großen Klassen von Beschreibungsystemen, die wir im Gebiet der formalen Sprachen behandeln, sind Automaten und Grammatiken. Bei endlichen Automaten, dem zu regulären Grammatiken äquivalenten Automatenmodell, hat sich die Anzahl der Zustände als Maß für die Komplexität und Größe durchgesetzt. Das ist als erster Ansatz naheliegend, da ein endlicher Automat nur aus einer Menge von Zuständen und Übergängen zwischen diesen Zuständen besteht, und die Anzahl der Übergänge durch die Anzahl der Zustände und die Größe des Eingabealphabets beschränkt ist. Noch aussagekräftigere Ergebnisse wären natürlich möglich, wenn die Anzahl der Übergänge miteinbezogen würde.

Bei Kellerautomaten, dem zu kontextfreien Grammatiken äquivalenten Automatenmodell, wird das Problem, ein sinnvolles Maß zu finden, schon schwieriger. Kellerautomaten bestehen aus einer endlichen Menge von Zuständen, einer endlichen Menge von Kellersymbolen und einer endlichen Menge von Übergängen. Bei diesen Übergängen wird nicht nur der Zustand gewechselt, sondern es werden auch auf einem theoretisch unendlichen Speicher, dem Keller, Kellersymbole gelesen und gespeichert. Neben der absoluten Länge der Beschreibung eines solchen Automaten spielen für die Komplexität sowohl die Anzahl der Zustände als auch die Anzahl der Kellersymbole eine Rolle [GPW82], [GPW93]. Einen guten Überblick über weitere Aspekte der Beschreibungskomplexität bei Automatenmodellen bietet [GKK+02].

Betrachtet man nun kontextfreie Grammatiken und insbesondere Grammatiksysteme, bietet sich eine noch größere Vielzahl von Möglichkeiten an, die Größe und besonders die Komplexität eines solchen Modells zu beschreiben.

Zunächst einmal steigt die Komplexität einer kontextfreien Grammatik sicherlich, wenn sich die Anzahl der Nichtterminale erhöht. In [Gru67] wird dieses Maß betrachtet und gezeigt, daß für jede natürliche Zahl n eine Sprache existiert, für deren Erzeugung eine kontextfreie Grammatik mindestens n Nicht-

terminale benötigt. Das gilt sogar für Sprachen über binären Alphabeten, allerdings gilt es nicht im unären Fall.

[GPW82] zieht einen Vergleich zwischen der Beschreibungskomplexität von kontextfreien Grammatiken und Kellerautomaten. Hier wird gezeigt, daß es für jedes Paar von natürlichen Zahlen n und p eine Sprache gibt, die von einem Kellerautomaten mit n Zuständen und p Kellersymbolen akzeptiert wird. Jede kontextfreie Grammatik aber, die diese Sprache erzeugt, braucht mindestens n^2p Nichtterminale. Dieses Ergebnis setzt jedoch nicht nur die zwei Beschreibungssysteme für kontextfreie Sprachen in Relation zueinander. Darüber hinaus kann man damit auch noch zeigen, daß die Tripelkonstruktion (siehe [HU79]), eine Konstruktion zur Umwandlung von Kellerautomaten in kontextfreie Grammatiken, von der Größe her optimale Ergebnisse liefert.

Ein ähnlich naheliegendes Maß für die Komplexität einer kontextfreien Grammatik wie die Anzahl der Nichtterminale ist die Anzahl der Produktionen. Auch hier kann gezeigt werden, daß es für jede natürliche Zahl n eine Sprache gibt, für deren Erzeugung eine kontextfreie Grammatik mindestens n Produktionen benötigt [Gru69]. Das gilt für ein beliebiges Alphabet. Einen tieferen Einblick in die Erforschung der eben erwähnten Maße und die Behandlung verschiedener damit zusammenhängenden Fragen bietet [Gru73].

Es gibt also unendliche Hierarchien von Sprachklassen über die Anzahl der Nichtterminale beziehungsweise die Anzahl der Produktionen von kontextfreien Grammatiken. Die Einschränkung dieser Parameter hat direkte Auswirkung auf die generative Mächtigkeit. Also erfassen die Maße Anzahl der Produktionen und Anzahl der Nichtterminale wichtige Aspekte der Komplexität einer kontextfreien Grammatik.

Allerdings entsprechen diese Maße nicht der intuitiven Auffassung von Größe. Wenn man die Größe oder Länge einer Beschreibung über einem endlichen Alphabet messen will, erwartet man, daß es für eine Länge n nur endlich viele Beschreibungen gibt. Es gibt zum Beispiel für ein festes n und ein festes Eingabealphabet nur endlich viele Wörter der Länge n oder nur endlich viele endliche Automaten mit n Zuständen. Im Gegensatz dazu kann man für einen bestimmten Wert n unendlich viele verschiedene Grammatiken angeben, die n Produktionen beziehungsweise n Nichtterminale haben. Das kommt dadurch zustande, daß die rechte Seite von kontextfreien Produktionen aus beliebig vielen Terminalen und Nichtterminalen bestehen kann und so die Größe der Beschreibung durch die Anzahl der Produktionen oder Nichtterminale nicht beschränkt ist.

Wir wollen uns in dieser Arbeit aber auch mit Maßen beschäftigen, die die Größe einer Grammatik im Sinne der Länge der Beschreibung darstellen. Es liegt nahe, dafür die Anzahl der Symbole zu zählen, die man braucht, um die Grammatik zu beschreiben.

Vergleicht man jetzt eine kontextfreie Grammatik mit einem endlichen Automaten, so kann man für jede rekursive Funktion f und eine beliebige natürliche Zahl n eine reguläre Sprache finden, für deren Beschreibung ein endli-

cher Automat $f(n)$ Zustände braucht. Diese Sprache kann aber auch von einer kontextfreien Grammatik der Größe (im eben definierten Sinne) n beschrieben werden [MF71]. Das Ergebnis ist nicht nur für unsere Arbeit wichtig, weil ein Größenmaß benutzt wurde, das der intuitiven Auffassung der Länge einer Beschreibung entspricht. Es zeigt auch, daß kontextfreie Grammatiken manche regulären Sprachen besser beschreiben können als endliche Automaten. Die Einsparung, die man durch eine kontextfreie Grammatik erreichen kann, ist dabei durch keine rekursive Funktion beschränkt. Dieses Phänomen wird auch nichtrekursiver Tradeoff genannt.

Eine weitere Möglichkeit, die Größe einer kontextfreien Grammatik zu beschreiben, hat man, wenn man das vorher beschriebene Maß der Anzahl der Nichtterminale beibehält, aber die Struktur der Grammatik durch eine geeignete Normalform einschränkt. Es bietet sich z.B. die Chomskynormalform (siehe [HU79]) an, bei der die rechte Seite von Produktionen entweder aus einem Terminal oder aus zwei Nichtterminalen besteht. Wenn man die Anzahl der Nichtterminale auf eine natürliche Zahl n beschränkt, gibt es nur endlich viele kontextfreie Grammatiken in Chomskynormalform mit n Nichtterminalen. Kontextfreie Grammatiken in Chomskynormalform wurden bezüglich ihrer Beschreibungskomplexität in [Pir75] untersucht.

Der Vollständigkeit halber sei erwähnt, daß auch die Greibach-Normalform in [PK75] untersucht wurde. Diese strukturelle Beschränkung erlaubt jedoch nicht, das Maß der Anzahl der Variablen als echtes Größenmaß zu betrachten.

Auch im Falle der Grammatiksysteme können wir sehen, daß mit der Anzahl der Produktionen beziehungsweise Nichtterminale die Komplexität der Beschreibung steigt. Die beiden Maße Anzahl der Nichtterminale und Anzahl der Produktionen wurden in [DP91] und [BR02] dazu verwendet, CD Grammatiksysteme und kontextfreie Grammatiken zu vergleichen und in [Pău92], um parallele Grammatiksysteme und kontextfreie Grammatiken zu vergleichen. In beiden Fällen erhält man das Ergebnis, daß Grammatiksysteme in einigen Fällen kontextfreie Sprachen effektiver beschreiben können als kontextfreie Grammatiken. Die Einsparung ist durch keine Funktion beschränkt. Darüber hinaus kann für die Länge der Beschreibung ein ähnlich gutes Ergebnis erzielt werden.

Allerdings wird die Einfachheit einer Beschreibung bei CD Grammatiksystemen auch durch andere Faktoren bestimmt. Ein CD Grammatiksystem besteht aus Komponenten, mit deren Anzahl auch die Komplexität der Beschreibung steigt. Außerdem ist nicht nur die gesamte Anzahl der Produktionen wichtig, sondern auch die Art und Weise, wie diese auf die Komponenten verteilt sind. Eine Komponente mit wenigen Produktionen ist sicher einfacher zu erfassen als eine Komponente mit vielen Produktionen. Es wird deshalb auch die maximale Anzahl von Produktionen in einer Komponente betrachtet. Für einige Ableitungsmodi ist es möglich, die Anzahl der Komponenten bzw. die maximale Anzahl der Produktionen mit einer Konstante zu beschränken und trotzdem alle Sprachen der entsprechenden Sprachklasse zu erzeugen [PDS94].

In dieser Arbeit wird uns als erstes die Frage beschäftigen, ob man für hy-

bride CD Grammatiksysteme ähnlich gute Ergebnisse erzielen kann wie für kontextfreie CD Grammatiksysteme. Teilweise können Ergebnisse über den Vergleich mit kontextfreien Grammatiken übertragen werden, da hybride CD Grammatiksysteme eine Generalisierung von kontextfreien CD Grammatiksystemen sind. Resultate über Maße, die einzelne Aspekte wie die Anzahl der Produktionen in einer Komponente beleuchten, müssen aber unabhängig von den Ergebnissen bei kontextfreien CD Grammatiksystemen erzielt werden.

Resultate über die Beschreibungskomplexität von CD Grammatiksystemen im Zusammenhang mit Maßen, die den Anforderungen, die man intuitiv an ein Größenmaß stellt, entsprechen, sind nur im Vergleich mit kontextfreien Grammatiken bekannt. Tradeoffs zwischen verschiedenen Arten von CD Grammatiksystemen sind nicht bekannt. Das kann daran liegen, daß für kontextfreie CD Grammatiksysteme nur wenige Hilfsmittel zur Verfügung stehen, mit denen man Aussagen darüber machen kann, ob die erzeugte Sprache in einer bestimmten Sprachklasse enthalten ist oder nicht.

Ein wichtiges Ziel dieser Arbeit ist es deshalb, auch eingeschränkte kontextfreie CD Grammatiksysteme, die aber die positiven Eigenschaften des uneingeschränkten Modells möglichst beibehalten, auf ihre Beschreibungskomplexität hin zu untersuchen.

Eine naheliegende Art, eine Grammatik zu vereinfachen, ist eine Einschränkung der Struktur ihrer Produktionen. Wir werden zunächst die metalinearen CD Grammatiksysteme analog zu metalinearen kontextfreien Grammatiken definieren. Das heißt, daß die rechte Seite einer Startproduktion mehrere Nichtterminale enthalten darf, alle anderen Produktionen aber jeweils höchstens ein Nichtterminal auf der rechten Seite haben.

Es stellt sich nun die Frage, ob das so erhaltene Modell auch nicht-kontextfreie Sprachen erzeugen kann. Außerdem sollte untersucht werden, ob man Aussagen über die Beschreibungskomplexität machen kann, die sich auf Maße beziehen, die die Größe oder Länge einer Beschreibung wiedergeben. Interessant wäre dabei nicht nur das Verhältnis der neu definierten Klassen untereinander, sondern auch das Verhältnis zu kontextfreien Sprachen.

Eine weitere Möglichkeit Grammatiken einzuschränken, ist der endliche Index. Hier ist die Anzahl der Nichtterminale in einer Satzform während der Erzeugung eines Wortes durch eine Konstante beschränkt. Diese Einschränkung auf kontextfreie CD Grammatiksysteme ist nicht ganz so stark wie die Metalinearität und wurde in [DP90] zum ersten Mal eingeführt. Es bleibt zu untersuchen, ob man für dieses Modell ähnlich gute Ergebnisse in der Beschreibungskomplexität erzielen kann wie für metalineare CD Grammatiksysteme.

1.2 Übersicht

Die weiteren Kapitel sind wie folgt aufgebaut:

- Kapitel 2 umfaßt die Grundlagen aus der Mathematik und der Theorie der formalen Sprachen, die für diese Arbeit benötigt werden. Wir führen die benutzten Notationen ein und definieren vor allen Dingen die Grammatikmodelle, die die Sprachklassen der Chomskyhierarchie erzeugen. Einige Automatenmodelle, die wir später benötigen, werden ebenfalls eingeführt. Weiterhin stellen wir das Gebiet der Beschreibungskomplexität vor und befassen uns mit Komplexitätsmaßen für Grammatiken und Grammatiksysteme.

- In Kapitel 3 stellen wir einige den Grammatiksystemen verwandte neuere Modelle aus der Theorie der formalen Sprachen vor. Außerdem fassen wir die Resultate über diese Modelle zusammen, die wir in den folgenden Kapiteln benutzen werden.

- Kapitel 4 enthält wichtige Ergebnisse auf dem Gebiet der kontextfreien CD Grammatiksysteme. Nach den nötigen Definitionen wird die Funktionsweise von CD Grammatiksystemen anhand einiger Beispiele verdeutlicht. Im nächsten Abschnitt fassen wir die Ergebnisse über die generative Mächtigkeit von kontextfreien CD Grammatiksystemen in den verschiedenen Ableitungsmodi zusammen. Der letzte Abschnitt enthält einen Überblick über die Resultate für kontextfreie CD Grammatiksysteme im Bereich Beschreibungskomplexität.

- Kapitel 5 befaßt sich mit den hybriden CD Grammatiksystemen. Für diese Generalisierung von kontextfreien CD Grammatiksystemen geben wir zuerst alle nötigen Ergebnisse an und fassen dann die Ergebnisse über die generative Mächtigkeit von hybriden CD Grammatiksystemen zusammen. Danach folgt ein Abschnitt über die Beschreibungskomplexität von hybriden CD Grammatiksystemen. Dabei werden Resultate gezeigt, die die Anzahl der Komponenten und die maximale Anzahl der Produktionen in einer Komponente nach oben beschränken.

- Nachdem wir kontextfreie CD Grammatiksysteme und auch hybride CD Grammatiksysteme untersucht haben, werden wir im Kapitel 6 metalineare CD Grammatiksystemen definieren. Danach wird die generative Mächtigkeit dieses Modells untersucht. Wir zeigen, daß die Klasse der m-linearen CD Grammatiksysteme mit der Klasse der m-linearen ET0L-Systeme und Matrixgrammatiken zusammenfällt. Es wird sich erweisen, daß man auch mit diesen eingeschränkten CD Grammatiksystemen nicht-kontextfreie Sprachen erzeugen kann. Außerdem werden wir ein Pumpinglemma für diese Sprachklasse beweisen und Abschlußeigenschaften untersuchen. Mit Hilfe dieser Ergebnisse erhalten wir dann eine unendliche Hierarchie über der Breite von metalinearen CD Grammatiksystemen. Der nächste Abschnitt befaßt sich mit der Beschreibungskomplexität von metalinearen CD Grammatiksystemen. Es wird gezeigt, daß gerade $(2m-1)$-lineare Sprachen von m-linearen CD Grammatiksystemen erzeugt werden können, aber nicht $2m$-lineare Sprachen. Durch Anwendung dieser Eigenschaft erhalten wir nichtrekursive Tradeoffs zwischen $(m+1)$- und m-linearen CD Grammatiksystemen und auch zwi-

schen m-linearen CD Grammatiksystemen und $(2m-1)$-linearen kontextfreien Grammatiken.

- Kapitel 7 befaßt sich mit CD Grammatiksystemen von endlichem Index. Nach den Definitionen werden die Ergebnisse auf dem Gebiet der generativen Mächtigkeit zusammengefaßt. Ähnlich wie bei metalinearen CD Grammatiksystemen erhalten wir eine Klasse, die mit der Klasse der ET0L-Systeme und Matrixgrammatiken von endlichem Index zusammenfällt. Auch hier können wir weiterhin nicht-kontextfreie Sprachen erzeugen. Die Klasse der von metalinearen CD Grammatiksystemen erzeugten Sprachen ist echt enthalten. Außerdem gibt es ein Pumpinglemma und es existiert eine unendliche Hierarchie über den Index von CD Grammatiksystemen. Im nächsten Abschnitt zeigen wir Ergebnisse auf dem Gebiet der Beschreibungskomplexität. Auch in diesem Fall können nichtrekursive Tradeoffs bewiesen werden. Zwischen CD Grammatiksystemen vom Index $m+1$ und vom Index m und auch zwischen CD Grammatiksystemen vom Index m und m-linearen CD Grammatiksystemen besteht ein solcher nichtrekursiver Tradeoff.
- Kapitel 8 schließlich faßt die Arbeit und die erzielten Resultate zusammen. Weiterhin werden ein Ausblick für zukünftige Arbeit auf dem Gebiet gegeben und offene Fragen aufgelistet.

Die Mehrzahl der in dieser Arbeit zusammengefaßten Ergebnisse ist bereits veröffentlicht oder zur Veröffentlichung akzeptiert worden. Die Resultate über die Beschreibungskomplexität von hybriden CD Grammatiksystemen aus Kapitel 5 wurden in [Sunb] beziehungsweise [Sun03] gezeigt. Die Definition und Resultate über die generative Mächtigkeit von metalinearen CD Grammatiksystemen finden sich in [Sun07]. Die weiteren Ergebnisse aus dem sechsten Kapitel, die die Beschreibungskomplexität von metalinearen CD Grammatiksystemen betreffen, wurden in [Sun05b] gezeigt. Kapitel 7 hat Ergebnisse über die Beschreibungskomplexität von CD Grammatiksystemen von endlichem Index zum Inhalt und basiert auf [Sun05a] beziehungsweise [Suna].

Kapitel 2

Grundlagen

Dieses Kapitel umfaßt die für die Arbeit benötigten mathematischen Grundlagen und grundlegende Konzepte aus der Theorie der formalen Sprachen. Es dient vor allen Dingen dazu, die in den folgenden Kapiteln verwendeten Notationen einzuführen und grundlegende Ergebnisse, die in den folgenden Kapiteln benutzt werden, zusammenzufassen. Weiterhin wird das Gebiet der Beschreibungskomplexität vorgestellt.

Die Definitionen in diesem Kapitel richten sich weitgehend nach dem Standardwerk [HU79]. Das Konzept der linearen und metalinearen Grammatiken stammt ursprünglich aus [CS63] wir richten uns jedoch nach der Schreibweise in [BB93].

2.1 Mathematische Grundlagen

Die Menge der natürlichen Zahlen wird mit $\mathbb{N}$, die Menge der reellen Zahlen mit $\mathbb{R}$ und die Menge der positiven reellen Zahlen mit $\mathbb{R}^+$ bezeichnet. Außerdem ist $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$.

Für die Kardinalität einer endlichen Menge U schreiben wir $|U|$ und für die Potenzmenge $P(U)$. Das Zeichen $\emptyset$ repräsentiert die leere Menge. Bei einer Menge U bezeichnet $\min(U)$ das kleinste und $\max(U)$ das größte Element, wenn ein solches Element existiert. Weiterhin schreiben wir $\sup(U)$ für das Supremum von U. Seien U und V zwei Mengen. Sie heißen äquivalent, wenn $U = V$. Mit $U \subset V$ bezeichnen wir, daß U eine echte Teilmenge von V ist. Wenn $U = V$ oder $U \subset V$ gilt, dann schreiben wir $U \subseteq V$. Mit $U \setminus V$ wird die Differenz von U und V also die Menge $\{x \mid x \in U, x \notin V\}$ bezeichnet.

Definition 2.1 *Sei U eine Menge. Ein System $V = \{V_1, \ldots, V_n\}$ mit $n \geq 1$ aus Teilmengen von U heißt Partition von U, falls*

- $\bigcup_{i=1}^{n} V_i = U$,
- $V_i \cap V_j = \emptyset$ *für alle* $1 \leq i \neq j \leq n$ *und*
- $\emptyset \notin V$.

Definition 2.2 *Das **Kreuzprodukt** zweier beliebiger Mengen U und V ist de-*

finiert durch

$$U \times V = \{(a, b) \mid a \in U,\, b \in V\}.$$

Eine ***(binäre) Relation*** r *von* U *und* V *ist eine beliebige Teilmenge des Kreuzprodukts*

$$r \subseteq U \times V.$$

Falls $r \subseteq U \times U$ *gilt, wird* r *Relation auf* U *genannt.*

Seien U, V, W *beliebige Mengen. Als Komposition* $\circ$ *zweier Relationen* $r \subseteq U \times V$ *und* $s \subseteq V \times W$ *bezeichnen wir die Relation*

$$r \circ s = \{(a, c) \in U \times W \mid \text{ es gibt ein } b \in V \text{ mit } (a, b) \in r \text{ und } (b, c) \in s\}.$$

Definition 2.3 *Eine Relation* $f \subseteq U \times V$ *heißt* ***partielle Funktion*** *von* U *nach* V*, wenn für alle* $a \in U$ *und* $b_1, b_2 \in V$ *folgendes gilt:*

Wenn $(a, b_1) \in f$ *und* $(a, b_2) \in f$*, dann ist* $b_1 = b_2$.

Eine partielle Funktion f *heißt* ***(totale) Funktion*** *oder* ***Abbildung*** *von* U *nach* V*, wenn sie folgende Eigenschaft hat:*

Für alle $a \in U$ *gibt es ein* $b \in V$ *mit* $(a, b) \in f$.

Statt $f \subseteq U \times V$ *und* $(a, b) \in f$ *schreiben wir im Fall von partiellen Funktionen auch* $f : U \to V$ *und* $f(a) = b$.

Das ***Urbild*** *einer Teilmenge* M *von* V *bezüglich einer Funktion* $f : U \to V$ *ist*

$$f^{-1}(M) = \{a \in U \mid f(a) = b \text{ und } b \in M\}.$$

Die Funktion f *ist injektiv, wenn für alle* $b \in V$ *höchstens ein* $a \in U$ *existiert mit* $f(a) = b$*. Außerdem heißt* f *surjektiv, wenn für alle* $b \in V$ *mindestens ein* $a \in U$ *mit* $f(a) = b$ *existiert. Wenn* f *sowohl injektiv als auch surjektiv ist, nennt man sie* ***bijektiv****.*

Definition 2.4 *Ein* ***gerichteter Graph*** $G = (V, E)$ *besteht aus einer endlichen Menge von Knoten* V *und einer endlichen Menge von geordneten Paaren in* $V \times V$*, den gerichteten Kanten.*

Ein ***Pfad*** *ist eine Sequenz von Knoten* $v_1, \ldots, v_k$ *mit* $k \geq 1$*, so daß es Kanten* (v_i, v_{i+1}) *mit* $1 \leq i \leq k - 1$ *gibt.*

Wenn (v_i, v_{i+1}) *eine Kante in* G *ist, dann nennt man* v_i *Vorgänger von* v_{i+1} *und* v_{i+1} *Nachfolger von* v_i.

Ein ***geordneter Baum*** *ist ein gerichteter Graph mit den folgenden Eigenschaften:*

1. *Es gibt einen Knoten, der keine Vorgänger hat. Er wird* ***Wurzel*** *genannt.*
2. *Jeder Knoten außer der Wurzel hat genau einen Vorgänger.*

3. *Die Nachkommen eines Knotens sind von links nach rechts geordnet.*

Es gibt eine spezielle Terminologie für Bäume, die von der für Graphen abweicht. Der Nachkomme eines Knotens wird ***Sohn*** *und der Vorgänger* ***Vater*** *genannt. Ein Knoten ohne Nachkommen heißt* ***Blatt*** *und alle anderen Knoten sind* ***innere Knoten****.*

2.2 Formale Sprachen

Definition 2.5 *Ein* ***Alphabet*** Σ *ist eine nichtleere endliche Menge, deren Elemente* ***Zeichen*** *oder* ***Symbole*** *genannt werden. Ein* ***Wort*** $w = (a_1, a_2, \ldots, a_k)$ *über dem Alphabet* Σ *ist eine endliche Folge von Symbolen aus* Σ*. Dabei ist* k *die Länge des Wortes* w *und wir schreiben* $|w| = k$*. Die Notation* $w = (a_1, a_2, \ldots, a_k)$ *vereinfachen wir folgendermaßen:* $w = a_1 a_2 \ldots a_k$*. Mit* ε *wird das Wort der Länge* 0 *oder auch leere Wort bezeichnet. Außerdem ist* $w^R = a_k a_{k-1} \ldots a_1$ *die Spiegelung des Wortes* w*.*

Sei Σ *ein Alphabet, dann ist* Σ^* *die Menge aller Wörter über* Σ *und* $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$*. Außerdem ist für* $k \geq 0$

$$\Sigma^k = \{w \in \Sigma^* \mid |w| = k\}$$

und

$$\Sigma^{\leq k} = \{w \in \Sigma^* \mid |w| \leq k\}.$$

Definition 2.6 *Sei* Σ *ein Alphabet. Die* ***Konkatenation*** *ist eine Abbildung* $\cdot : \Sigma^* \times \Sigma^* \to \Sigma^*$*. Es seien* $v = a_1 \ldots a_k \in \Sigma^*$ *und* $w = b_1 \ldots b_l \in \Sigma^*$*,* $k, l \geq 0$*, dann ist*

$$v \cdot w = a_1 \ldots a_k b_1 \ldots b_l.$$

Zur Vereinfachung schreiben wir statt $v \cdot w$ *auch* vw*.*

Definition 2.7 *Eine* ***(formale) Sprache*** L *über dem Alphabet* Σ *ist eine beliebige Teilmenge von* Σ^**. Außerdem bezeichnen wir mit* $alph(L)$ *das kleinste Alphabet* Σ' *mit* $L \subseteq \Sigma'^*$*.*

Definition 2.8 *Sei* Σ *ein Alphabet. Die* ***Konkatenation von zwei Sprachen*** $L_1, L_2 \subseteq \Sigma^*$ *definieren wir mit*

$$L_1 \cdot L_2 = \{vw \mid v \in L_1, w \in L_2\}.$$

Außerdem setzen wir für eine Sprache $L \in \Sigma^*$

$$L^0 = \{\varepsilon\}, \qquad L^k = L^{k-1} \cdot L,\ k \geq 1, \qquad L^* = \bigcup_{k \geq 0} L^k.$$

Definition 2.9 *Seien Σ_1 und Σ_2 zwei Alphabete. Eine Abbildung $h : \Sigma_1 \to \Sigma_2$ mit der Eigenschaft*

$$h(u \cdot v) = h(u) \cdot h(v)$$

*für alle $u, v \in \Sigma_1^*$ nennen wir **Homomorphismus**. Aus der Definition folgt, daß $h(\varepsilon) = \varepsilon$ gilt.*

*Ein Homomorphismus h heißt **Isomorphismus**, wenn h bijektiv ist und h^{-1} auch ein Homomorphismus ist.*

Homomorphismen kann man für viele algebraische Strukturen wie Vektorräume, Gruppen, Ringe und Körper in ähnlicher Weise definieren. Gibt es einen Isomorphismus zwischen zwei Strukturen, so nennt man diese isomorph.

Sei Σ ein Alphabet und $N \subseteq \Sigma$. Sei weiterhin $\alpha \in \Sigma^*$. Mit $|\alpha|_N$ bezeichnen wir die Anzahl der Symbole aus N in α.

2.2.1 Grammatiken

Zur Erzeugung von formalen Sprachen benutzen wir in dieser Arbeit vor allem Grammatiken. Eine Grammatik besteht aus zwei disjunkten endlichen Mengen von Symbolen, den Terminalen und den Nichtterminalen. Weiterhin gibt es eine endliche Menge von Ersetzungsregeln oder Produktionen. Mit einer Grammatik wird ein Wort erzeugt, indem mit einem Startsymbol anfangend Ersetzungsregeln angewendet werden. Dabei ersetzt man die linke Seite der Regel durch die rechte Seite. Die Wörter bestehend aus Terminalen, die so erreichen werden, bilden die von der Grammatik erzeugte Sprache.

Es gibt viele Formen von Grammatiken, die sich vor allen Dingen durch die Art ihrer Ersetzungsregeln unterscheiden. Zunächst wird eine Grammatik wie folgt definiert:

Definition 2.10 *Eine **(formale) Grammatik** ist ein Viertupel*

$$G = (N, T, P, S),$$

wobei

1. *N eine endliche Menge von Nichtterminalen,*
2. *T eine endliche Menge von Terminalen mit $N \cap T = \emptyset$,*
3. *P eine endliche Menge von Produktionen aus $(N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$ und*
4. *S ein ausgezeichnetes Nichtterminal, das Startsymbol, ist.*

Für $(\alpha, \beta) \in P$ schreiben wir meist $\alpha \to \beta$.

Mit dieser Grammatik kann man nun mit dem Startsymbol beginnend Wörter erzeugen. Dabei wendet man solange Produktionen an, bis ein Wort aus Terminalen erzeugt wurde. Eine solche Folge von Ersetzungen ist dann eine Ableitung. Formal definieren wir die Ableitungsrelation wie folgt:

Definition 2.11 *Sei $G = (N, T, P, S)$ eine Grammatik. Zunächst definieren wir die **Ableitungsrelation** $\Rightarrow$ auf $(N \cup T)^*$. Es gilt*

$$\begin{aligned}
\alpha \Rightarrow \beta \quad & \text{genau dann, wenn } \alpha = \gamma_1\alpha_1\gamma_2, \beta = \gamma_1\beta_1\gamma_2 \text{ und } \alpha_1 \to \beta_1 \in P \\
& \text{mit } \gamma_1, \gamma_2, \alpha_1, \beta_1 \in (N \cup T)^* \text{ ist,} \\
\alpha \overset{0}{\Longrightarrow} \beta \quad & \text{genau dann, wenn } \alpha = \beta \text{ ist,} \\
\alpha \overset{n+1}{\Longrightarrow} \beta \quad & \text{genau dann, wenn } \alpha \overset{n}{\Longrightarrow} \alpha' \text{ und } \alpha' \Rightarrow \beta \\
& \text{mit } \alpha' \in (N \cup T)^* \text{ ist,} \\
\alpha \overset{*}{\Longrightarrow} \beta \quad & \text{genau dann, wenn } \alpha \overset{n}{\Rightarrow} \beta \text{ für ein } n \geq 0 \text{ ist.}
\end{aligned}$$

Definition 2.12 *Die von einer Grammatik $G = (N, T, P, S)$ erzeugte **Sprache** ist*

$$L(G) = \{w \mid w \in T^*, S \overset{*}{\Rightarrow} w\}.$$

*Eine Zeichenkette $\alpha \in (N \cup T)^*$ heißt **Satzform**, falls $S \overset{*}{\Rightarrow} \alpha$ gilt. Die von G erzeugte Sprache ist also die Menge der Satzformen, die nur aus Terminalsymbolen bestehen.*

Definition 2.13 *Man nennt zwei Grammatiken G_1 und G_2 **(schwach) äquivalent**, wenn $L(G_1) = L(G_2)$.*

Im Gegensatz zur Gleichheit der erzeugten Sprachen kann man den Begriff äquivalent auch im Sinne einer äquivalenten Struktur benutzen. Wenn also zwei Grammatiken dieselben Wörter erzeugen und jedem Wort ähnliche Ableitungsbäume zuordnen, die sich nur durch die Markierung der Knoten unterscheiden, nennt man sie strukturäquivalent ([Sal78]). Der Begriff äquivalent wird aber in dieser Arbeit nur im oben definierten Sinn benutzt.

Definition 2.14 *Sei $G = (N, T, P, S)$ eine Grammatik. Eine **Ableitung** in G von α nach β ist eine Folge $\alpha_1, \ldots, \alpha_k \in (N \cup T)^*$ mit $k \geq 1$, für die folgendes gilt:*

$$\alpha = \alpha_1, \qquad \beta = \alpha_k, \qquad \alpha_i \Rightarrow \alpha_{i+1} \text{ für } 1 \leq i \leq k-1.$$

In den folgenden Kapiteln werden wir bei einigen Konstruktionen einen Homomorphismus brauchen, mit dessen Hilfe man in einer Grammatik Nichtterminale markieren kann, während die Terminale nicht verändert werden.

Sei $G = (N, T, P, S)$ eine Grammatik, $i \geq 0$ und

$$N^{(i)} = \{A^{(i)} \mid A \in N\}.$$

Der Homomorphismus $(i) : N \cup T \to N^{(i)} \cup T$ ist folgendermaßen definiert:

$$\begin{aligned}
(i)(A) &= A^{(i)}, \; A \in N \text{ und} \\
(i)(a) &= a, \; a \in T.
\end{aligned}$$

Als nächstes führen wir verschiedene Arten von Grammatiken ein, die sich von der allgemeinen Form durch mehr oder weniger starke Einschränkungen unterscheiden. Wir werden sehen, daß alle diese Klassen von Grammatiken verschiedene ineinander echt enthaltene Sprachklassen erzeugen.

Definition 2.15 *Eine* ***reguläre Grammatik*** *ist ein Viertupel*

$$G = (N, T, P, S),$$

wobei

1. N *eine endliche Menge von Nichtterminalen,*
2. T *eine endliche Menge von Terminalen mit* $N \cap T = \emptyset$*,*
3. P *eine endliche Menge von Produktionen aus* $N \times T^* N^{\leq 1}$ *und*
4. S *das Startsymbol ist.*

Die Menge der regulären Grammatiken wird mit *REG* bezeichnet und die Menge der regulären Sprachen mit $\mathcal{L}(REG)$.

Die oben definierten regulären Grammatiken werden auch ***rechtslinear*** genannt. Analog werden Grammatiken mit Produktionen in $N \times N^{\leq 1} T^*$ ***linkslinear*** genannt. Beide Formen der regulären Grammatik sind äquivalent.

Lemma 2.1 *[HU79] Für jede linkslineare Grammatik* G *ist* $L(G)$ *eine reguläre Sprache.*

Außerdem kann man leicht die folgende Normalform zeigen.

Lemma 2.2 *Wenn* L *eine reguläre Sprache ist, dann gibt es eine linkslineare Grammatik* G_1 *mit Produktionen aus* $N \times (N^{\leq 1} T \cup \{\varepsilon\})$ *und* $L = L(G_1)$*. Außerdem gibt es auch eine rechtslineare Grammatik* G_2 *mit Produktionen aus* $N \times (T N^{\leq 1} \cup \{\varepsilon\})$ *und* $L = L(G_2)$*.*

Mit Hilfe dieser Normalform kann man besonders einfach die Äquivalenz der regulären Grammatiken zu den endlichen Automaten, die im nächsten Abschnitt definiert werden, beweisen.

Definition 2.16 *Eine* ***kontextfreie Grammatik*** *ist ein Viertupel*

$$G = (N, T, P, S),$$

wobei

1. N *eine endliche Menge von Nichtterminalen,*
2. T *eine endliche Menge von Terminalen mit* $N \cap T = \emptyset$*,*
3. P *eine endliche Menge von Produktionen aus* $N \times (N \cup T)^*$ *und*
4. S *das Startsymbol ist.*

Die Menge der kontextfreien Grammatiken wird mit *CF* bezeichnet und die Menge der kontextfreien Sprachen mit $\mathcal{L}(CF)$.

Die Ableitung einer kontextfreien Grammatik läßt sich auch mit Bäumen sehr gut darstellen. Dafür definieren wir jetzt den Begriff des Ableitungsbaums.

Definition 2.17 *Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik. Ein geordneter Baum ist ein **Ableitungsbaum**, wenn folgendes gilt:*

1. *Jeder Knoten hat eine **Markierung**, die ein Symbol aus $N \cup T \cup \{\varepsilon\}$ ist.*
2. *Die Wurzel hat die Markierung S.*
3. *Wenn ein innerer Knoten v mit A markiert ist, dann ist $A \in N$.*
4. *Wenn ein Knoten v die Markierung A hat und $v_1, \ldots, v_n$ die Söhne von v markiert mit $X_1, \ldots, X_n$ sind, dann ist $A \to X_1 \ldots X_n$ eine Produktion aus P.*
5. *Wenn ein Knoten mit ε markiert ist, dann ist er der einzige Sohn seines Vaters.*

Definition 2.18 *Eine kontextfreie Grammatik $G = (N, T, P, S)$ heißt **linear**, wenn jede Produktion in der Form $A \to uBv$ oder $A \to u$ ist, wobei $A, B \in N$ und $u, v \in T^*$ gilt. Eine Sprache L wird linear genannt, wenn es eine lineare Grammatik G mit $L(G) = L$ gibt.*

Die Menge der linearen Grammatiken bezeichnen wir mit *LIN* und die Menge der linearen Sprachen mit $\mathcal{L}(LIN)$.

Lemma 2.3 *[BB93] Zu jeder linearen Grammatik $G = (N, T, P, S)$ gibt es eine äquivalente Grammatik, die nur Produktionen der folgenden Form enthält*

$$A \to \alpha, \quad A \in N,\ \alpha \in T^{\leq 1}N^{\leq 1} \cup N^{\leq 1}T^{\leq 1} \cup \{\varepsilon\}.$$

Definition 2.19 *Eine kontextfreie Grammatik $G = (N, T, P, S)$ heißt **m-linear**, wenn jede Produktion in der Form*

$$S \to A_1 \ldots A_{m_0}, \qquad A \to uBv \qquad \text{oder} \qquad A \to u$$

*ist, wobei $A, A_1, \ldots, A_{m_0}, B \in (N \setminus \{S\})$, $m_0 \leq m$, $m \geq 1$ und $u, v \in T^*$ sind. Eine Sprache L wird m-linear genannt, wenn eine m-lineare Grammatik G mit $L(G) = L$ existiert. Ist eine Grammatik oder Sprache m-linear für ein $m \geq 1$, dann nennt man sie auch **metalinear**.*

Die Menge der m-linearen Grammatiken wird mit *mLIN* bezeichnet. Außerdem ist $METALIN = \bigcup_{m \geq 1} mLIN$. Die entsprechenden Sprachklassen nennen wir $\mathcal{L}(mLIN)$ und $\mathcal{L}(METALIN)$.

Wir werden uns besonders in Kapitel 6 mit metalinearen Grammatiken und Sprachen befassen und dabei die folgende Hierarchie brauchen.

Satz 2.1 *[BB93] Es gilt*

$$\begin{aligned}\mathcal{L}(LIN) \subset \mathcal{L}(2LIN) \subset \ldots \subset \mathcal{L}(mLIN) &\subset \\ \mathcal{L}((m+1)LIN) &\subset \mathcal{L}(METALIN) \subset \mathcal{L}(CF).\end{aligned}$$

Weitere Eigenschaften von linearen und metalinearen Sprachen, die wir im folgenden auch benutzen werden, wurden in [Gre66] bewiesen.

Lemma 2.4 *[Gre66] Sei Σ ein Alphabet mit $c \notin \Sigma$. Sei weiterhin $R \subseteq \Sigma^*$ und $S \subseteq \Sigma^*$ Es gilt $RcS \in \mathcal{L}(LIN)$ genau dann, wenn $R, S \in \mathcal{L}(LIN)$ ist und entweder $R \in \mathcal{L}(REG)$ oder $S \in \mathcal{L}(REG)$ ist.*

Lemma 2.5 *[Gre66] Sei Σ ein Alphabet mit $c \notin \Sigma$. Sei weiterhin $S_i \subseteq \Sigma^*$ für $1 \leq i \leq m$ und $m \geq 2$. Wenn $S_1cS_2c \ldots cS_m$ k-linear für ein $k > 1$ ist und $S_1 \notin \mathcal{L}(REG)$ gilt, dann ist $S_2c \ldots cS_m$ höchstens $(k-1)$-linear.*

Lemma 2.6 *[Gre66] Sei Σ ein Alphabet mit $c \notin \Sigma$. Sei weiterhin $S_i \subseteq \Sigma^*$ für $1 \leq i \leq m$ und $m \geq 2$. Wenn $S_1cS_2c \ldots cS_m$ k-linear für ein $k > 1$ ist, dann sind höchstens k der S_i nicht in $\mathcal{L}(REG)$.*

Wir kommen jetzt zu einer weiteren Möglichkeit, die Form von kontextfreien Grammatiken einzuschränken, nämlich dem Begriff des endliche Index. Auch hier erhält man eine echte Teilmenge der kontextfreien Sprachen. Wir werden den Definitionen aus [Sal78] folgen, da sie zu den für CD Grammatiksystemen üblichen Definitionen äquivalent sind. Es sei jedoch auch erwähnt, daß z.B. in [BB93] der endliche Index für kontextfreie Grammatiken als eine stärkere Einschränkung definiert ist und auch zu einer kleineren Sprachklasse führt.

Definition 2.20 *Sei $G = (N, T, P, S)$ ein kontextfreie Grammatik. Für die Ableitung*

$$D : \alpha_1 \Rightarrow \ldots \Rightarrow \alpha_k$$

von G mit $\alpha_j \in (N \cup T)^*$, $1 \leq j \leq k$ ist der Index von D definiert durch

$$Ind(D) = \max\{|\alpha_j|_N \mid 1 \leq j \leq k\}.$$

Für ein Wort $w \in L(G)$ definieren wir den Index mit

$$Ind(w) = \min\{Ind(D) \mid D : S \stackrel{*}{\Rightarrow} w\}.$$

*Der **Index von G** ist*

$$Ind(G) = \sup\{Ind(w) \mid w \in L(G)\}.$$

Eine kontextfreie Sprache L wird ***vom Index m*** genannt, wenn eine Grammatik G vom Index m mit $L(G) = L$ existiert. Wenn eine Grammatik oder Sprache vom Index m für ein $m \geq 1$ ist, nennt man sie auch ***von endlichem Index***.

Die Klasse der kontextfreien Grammatiken vom Index m wird mit *mFIN* bezeichnet und die Klasse der kontextfreien Grammatiken von endlichem Index mit *FIN*. Für entsprechenden Sprachklassen schreiben wir $\mathcal{L}(mFIN)$ und $\mathcal{L}(FIN)$.

Die nächste Klasse von Grammatiken wird die Klasse der kontextsensitiven Sprachen erzeugen. Diese Sprachklasse brauchen wir vor allen Dingen zur Einordnung der generativen Mächtigkeit der untersuchten CD Grammatiksysteme.

Definition 2.21 *Eine* ***kontextsensitive Grammatik*** *ist ein Viertupel*

$$G = (N, T, P, S),$$

wobei

1. N *eine endliche Menge von Nichtterminalen,*
2. T *eine endliche Menge von Terminalen mit* $N \cap T = \emptyset$*,*
3. S *das Startsymbol ist,*
4. P *eine endliche Menge von Produktionen der Form*

   $$\alpha A \beta \to \alpha \gamma \beta \text{ mit } A \in N,\ \alpha, \beta \in (N \cup \Sigma)^*,\ \gamma \in (N \cup \Sigma)^+$$

   oder

   $$S \to \varepsilon$$

   ist. Kommt eine Regel in der zweiten Form vor, darf S *auf keiner rechten Seite einer Produktion sein.*

Die Menge der kontextsensitiven Grammatiken wird mit *CS* bezeichnet und die Menge der kontextsensitiven Sprachen mit $\mathcal{L}(CS)$.

Die am Anfang des Abschnitts definierten allgemeinen formalen Grammatiken werden auch als ***Typ-0-Grammatiken*** bezeichnet. Für die Menge der Typ-0-Grammatiken schreiben wir *RE* und für die Menge der daraus resultierenden rekursiv aufzählbaren Sprachen $\mathcal{L}(RE)$.

Ein fundamentales Ergebnis der Theorie der formalen Sprachen ist die Chomskyhierarchie. Es besagt, daß die Sprachklassen, die von den eben definierten Klassen von Grammatiken erzeugt werden, eine Hierarchie bilden. Darüber hinaus sind die Sprachklassen echt ineinander enthalten.

Satz 2.2 (Chomskyhierarchie) *Es gilt*

$$\mathcal{L}(REG) \subset \mathcal{L}(CF) \subset \mathcal{L}(CS) \subset \mathcal{L}(RE).$$

2.2.2 Automaten

Im letzten Abschnitt haben wir uns mit der Erzeugung von formalen Sprachen durch Grammatiken beschäftigt. Eine weitere wichtige Klasse von Beschreibungen für formale Sprachen sind Automaten. Man kann zeigen, daß es für jede Sprachklasse in der Chomskyhierarchie ein entsprechendes Automatenmodell gibt, das gerade diese Klasse von Sprachen akzeptiert. Da der Schwerpunkt dieser Arbeit auf Grammatiken liegt, erwähnen wir in diesem Abschnitt aber nur die Automatenmodelle, die im weiteren noch benutzt werden.

Zuerst wenden wir uns dem einfachsten Automatenmodell, das aus einer endlichen Menge von Zuständen und einer Übergangsfunktion besteht, zu. Die Menge von Zuständen wird auch endliche Kontrolle genannt, da man mit ihr eine endliche Menge von Informationen speichern kann.

Definition 2.22 *Ein **deterministischer endlicher Automat (DFA)** ist ein Fünftupel*

$$M = (Q, \Sigma, \delta, q_0, F),$$

wobei

1. *Q eine Menge von Zuständen,*
2. *Σ das Eingabealphabet,*
3. *$\delta : (Q \times \Sigma) \to Q$ die Übergangsfunktion,*
4. *$q_0 \in Q$ der Anfangszustand und*
5. *$F \subset Q$ die Menge der Endzustände ist.*

Ein DFA bearbeitet nun ein Wort aus Σ^*, indem er mit dem Startzustand beginnend jeweils ein Symbol liest und den Zustand entsprechend der Übergangsfunktion ändert.

Die Übergangsfunktion δ kann wie folgt auf Strings aus Σ^* erweitert werden:

$$\delta(q, \varepsilon) = q, \qquad \delta(q, wa) = \delta(\delta(q, w), a)$$

für alle $q \in Q$, $a \in \Sigma$, $w \in \Sigma^*$.

Ein Wort w aus Σ^* wird von einem deterministischen endlichen Automaten M ***akzeptiert***, wenn $\delta(q_0, w) = p \in F$. Die ***akzeptierte Sprache*** von M ist

$$T(M) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}.$$

Man kann zeigen, daß die Klasse der deterministischen endlichen Automaten gerade die Klasse der regulären Sprachen $\mathcal{L}(REG)$ akzeptiert. Deterministische endliche Automaten sind also das zu den rechtslinearen Grammatiken äquivalente Automatenmodell.

Die nächste Automatenklasse, die wir betrachten werden, besteht nicht nur aus einer Menge von Zuständen, also einer endlichen Kontrolle, sondern hat

auch einen sogenannten Keller, auf dem Symbole gespeichert und von dem sie wieder entfernt werden können. Dieser Keller ist in seiner Größe nicht beschränkt.

Definition 2.23 *Ein **Kellerautomat (PDA)** ist ein Siebentupel*

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F),$$

wobei

1. Q *eine Menge von Zuständen,*
2. Σ *das Eingabealphabet,*
3. Γ *das Kelleralphabet,*
4. δ *die Menge der Überführungsregeln, nämlich eine Abbildung von* $Q \times \Sigma \cup \{\varepsilon\} \times \Gamma$ *in endliche Teilmengen von* $Q \times \Gamma^*$*,*
5. $q_0 \in Q$ *der Anfangszustand,*
6. $Z_0 \in \Gamma$ *das Anfangskellersymbol,*
7. $F \subseteq Q$ *die Menge der Endzustände ist.*

Sei M ein PDA. Als ***Konfiguration*** von M bezeichnen wir Elemente aus $Q \times \Sigma^* \times \Gamma^*$. Diese Elemente stellen den jeweils aktuellen Zustand, die noch nicht gelesene Eingabe und den Inhalt des Kellers da. Ein Übergang oder Schritt führt der PDA durch, indem er eine Überführungsregel auf die momentane Konfiguration anwendet. Wenn

$$(p, B_1 \dots B_n) \in \delta(q, a, A)$$

für $a \in \Sigma^{\leq 1}$, $q, p \in Q$, $n \geq 0$ und $A, B_1, \dots, B_n \in \Gamma$, dann schreiben wir

$$(q, aw, A\alpha) \vdash_M (p, w, B_1 \dots B_n\alpha),$$

wobei $\alpha \in \Gamma^*$ und $w \in \Sigma^*$ ist.

Bei einem der eben beschriebenen Schritte wird also der Zustand von q nach p geändert und auf dem Keller das oberste Symbol A durch $B_1 \dots B_n$ ersetzt. Wenn $n = 0$ ist, wird das oberste Symbol auf dem Keller entfernt und die Kellerhöhe dadurch um ein Symbol verringert. Außerdem liest der PDA das Eingabezeichen $a \in \Sigma$, indem der sogenannte Eingabezeiger, der das aktuelle Zeichen der Eingabe anzeigt, auf der Eingabe um ein Symbol nach vorne gerückt wird. Falls $a = \varepsilon$ ist, bewegt sich der Eingabezeiger nicht.

Wenn eine Konfiguration (q', w', α') aus einer anderen Konfiguration (q, w, α) durch eine endliche Anzahl von Schritten resultiert (dabei sind null Schritte auch erlaubt), dann schreiben wir

$$(q, w, \alpha) \overset{*}{\vdash}_M (q', w', \alpha').$$

Die von M akzeptierte ***Sprache*** ist

$$T(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \overset{*}{\vdash}_M (p, \varepsilon, \varepsilon),\ p \in F\}.$$

Es ist möglich, den Begriff der von einem Kellerautomaten akzeptierten Sprache auch auf andere Weise zu definieren. Die erste Variante enthält alle Wörter, mit denen der Kellerautomat einen akzeptierenden Zustand erreicht:

$$T_f(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \overset{*}{\vdash}_M (p, \varepsilon, \beta),\ p \in F,\ \beta \in \Gamma^*\}.$$

Die zweite Variante vernachlässigt die Menge der akzeptierenden Zustände und enthält alle Wörter, mit denen der Kellerautomat seinen Keller vollständig leert:

$$T_\varepsilon(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \overset{*}{\vdash}_M (p, \varepsilon, \varepsilon),\ p \in Q\}.$$

Man kann zeigen, daß es für jeden PDA M einen PDA M' gibt, für den gilt, daß $T_\varepsilon(M) = T(M')$ ist. Ebenso kann man für jeden PDA M einen PDA M'' konstruieren, für den $T_f(M) = T(M'')$ ist.

Darüber hinaus akzeptieren Kellerautomaten gerade die Klasse der kontextfreien Sprachen $\mathcal{L}(CF)$, sie sind also die zu den kontextfreien Grammatiken äquivalente Automatenklasse. Diese Aussage beweist man mit Hilfe des Akzeptierungsbegriffs $T_\varepsilon(M)$.

In den folgenden Abschnitten brauchen wir allerdings einen eingeschränkten Kellerautomaten, der gerade die linearen Sprachen akzeptiert. Dazu definieren wir für einen Kellerautomaten $M = (\Sigma, Q, \Gamma, \delta, q_0, Z_0, F)$ den Begriff des Laufs. Ein ***Lauf*** von M ist eine Folge

$$(q_1, w_1, \alpha_1), (q_2, w_2, \alpha_2), \ldots, (q_k, w_k, \alpha_k)$$

mit

$$(q_i, w_i, \alpha_i) \vdash_M (q_{i+1}, w_{i+1}, \alpha_{i+1}),$$

$1 \leq i \leq k-1$, $q_1, \ldots q_k \in Q$, $w_1, \ldots, w_k \in \Sigma^*$, $\alpha_1, \ldots, \alpha_k \in \Gamma^*$. Bei einem ***akzeptierenden Lauf*** ist $q_1 = q_0$, $\alpha_1 = Z_0$, $q_k \in F$, $w_k = \varepsilon$ und $\alpha_k = \varepsilon$.

Definition 2.24 *Sei $M = (\Sigma, Q, \Gamma, \delta, q_0, Z_0, F)$ ein Kellerautomat. Ein Lauf von M $(q_1, w_1, \alpha_1), \ldots, (q_k, w_k, \alpha_k)$ heißt **1-turn-Lauf**, falls ein i mit $1 \leq i < k$ und den Eigenschaften*

$$|\alpha_1| \leq \ldots \leq |\alpha_{i-1}| \leq |\alpha_i|$$

und

$$|\alpha_i| > |\alpha_{i+1}| \geq \ldots \geq |\alpha_k|$$

existiert.

Ein 0-turn-Lauf hat die Eigenschaft, daß

$$|\alpha_j| \leq |\alpha_{j+1}|,\ 1 \leq j \leq k-1.$$

*M heißt **1-turn-Kellerautomat**, falls zu jedem Wort $w \in T(M)$ ein akzeptierender Lauf existiert, der ein 1-turn-Lauf oder ein 0-turn-Lauf ist.*

Satz 2.3 *[BB93] Die Klasse der 1-turn-Kellerautomaten akzeptiert gerade die Klasse der linearen Sprachen $\mathcal{L}(LIN)$.*

Als nächstes wenden wir uns dem Automatenmodell zu, das zu uneingeschränkten Grammatiken äquivalent ist. Es besteht wieder aus einer endlichen Menge von Zuständen und einer weiteren unendlichen Speichermöglichkeit. Diese wird Eingabeband oder einfach Band genannt und ist in Zellen unterteilt. Jede Zelle enthält ein Symbol. Nach rechts ist das Band unendlich, nach links jedoch beschränkt. Am Anfang der Bearbeitung eines Wortes ist die Turingmaschine im Anfangszustand und das Wort steht auf dem Band, wobei das erste Zeichen in der Zelle ganz links steht. Die Turingmaschine kann jetzt einen sogenannten Schreib-/Lesekopf auf dem Band hin und her bewegen und dabei Zeichen lesen und den Bandinhalt verändern.

Definition 2.25 *Eine Turingmaschine ist ein Siebentupel*

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F),$$

wobei

1. Q *eine Menge von Zuständen,*
2. $\Sigma \subset \Gamma \setminus \{B\}$ *das Eingabealphabet,*
3. Γ *das Bandalphabet,*
4. $\delta : (Q \times \Gamma) \to (Q \times \Gamma \times \{L, R\})$ *die (partielle) Übergangsfunktion,*
5. $B \in \Gamma$ *das Blanksymbol,*
6. $q_0 \in Q$ *der Anfangszustand und*
7. $F \subseteq Q$ *die Menge der Endzustände ist.*

Die ***Zustandsbeschreibung (ID)*** oder ***Konfiguration*** einer Turingmaschine M wird durch $\alpha_1 q \alpha_2$ dargestellt, wobei $\alpha_1, \alpha_2 \in \Gamma^*$ und $q \in Q$ ist. Dabei ist $\alpha_1\alpha_2$ der Bandinhalt bis zum am weitesten rechts stehenden und von B unterschiedlichen Bandsymbol. Weiterhin ist q der momentane Zustand, und der Schreib-/Lesekopf befindet sich auf dem ersten Symbol von α_2. Eine Konfiguration der Form $q_0 w$ mit $w \in \Sigma^*$ nennt man Anfangskonfiguration und eine Konfiguration der Form $\alpha_1 f \alpha_2$ mit $f \in F$ und $\alpha_1, \alpha_2 \in \Gamma^*$ heißt akzeptierende Konfiguration.

Die Turingmaschine kann jetzt von einer Konfiguration aus der Übergangsfunktion folgend das rechts neben ihrem Schreib-/ Lesekopf befindliche Symbol lesen, ein neues Symbol in diese Zelle schreiben und den Schreib-/ Lesekopf nach rechts oder nach links bewegen. Ein solcher Schritt oder ***Berechnungsschritt*** einer Turingmaschine wird wie folgt definiert. Sei

$$X_1 X_2 \ldots X_{i-1} q X_i \ldots X_n$$

mit $n \geq 0$ eine Konfiguration einer Turingmaschine M und sei weiterhin $\delta(q_i, X_i) = (p, Y, L)$, dann schreiben wir

$$X_1 X_2 \ldots X_{i-1} q X_i \ldots X_n \underset{M}{\vdash} X_1 X_2 \ldots X_{i-2} p X_{i-1} Y X_{i+1} \ldots X_n.$$

Wenn $\delta(q_i, X_i) = (p, Y, R)$, dann schreiben wir

$$X_1 X_2 \dots X_{i-1} q X_i \dots X_n \underset{M}{\vdash} X_1 X_2 \dots X_{i-1} Y p X_{i+1} \dots X_n.$$

Wenn für eine Turingmaschine M eine Zustandsbeschreibung ID' aus einer anderen Zustandsbeschreibung ID durch eine endliche Anzahl von Schritten resultiert (dabei sind null Schritte auch erlaubt), dann schreiben wir

$$ID \overset{*}{\vdash}_M ID'.$$

Wird ID' von ID aus in einem Schritt erreicht, dann nennen wir ID' ***Folgekonfiguration*** von ID.

Die von einer Turingmaschine M akzeptierte ***Sprache*** ist

$$T(M) = \{w \in \Sigma^* \mid q_0 w \overset{*}{\vdash}_M \alpha p \beta, p \in F, \alpha, \beta \in \Gamma^*\}.$$

Man kann zeigen, daß die Klasse der Turingmaschinen gerade die rekursiv aufzählbaren Sprachen $\mathcal{L}(RE)$ akzeptiert. Turingmaschinen sind daher das zu Typ-0-Grammatiken äquivalente Automatenmodell.

Wir sagen, daß eine Turingmaschine M auf einer Konfiguration ID ***anhält***, wenn von ID aus keine Folgekonfiguration erreicht werden kann. Eine weitere wichtige Sprachklasse ist die Klasse der Sprachen, die von Turingmaschinen akzeptiert werden, die auf allen Eingaben anhalten. Diese Klasse nennen wir die ***rekursiven*** Sprachen $\mathcal{L}(R)$ und es gilt $\mathcal{L}(R) \subset \mathcal{L}(RE)$.

Wir werden jetzt für jede Turingmaschine M eine Sprache definieren, deren Wörter gerade die Folgen von Konfigurationen sind, die mit einer Anfangskonfiguration beginnen, einer akzeptierenden Konfiguration enden und bei der eine Konfiguration durch eine Bewegung aus der vorherigen hervorgeht. Jedes Wort in dieser Sprache stellt also dar, wie M ein Wort akzeptiert.

Zur Definition der Sprache müssen wir M in eine bestimmte Form überführen. Dieses Resultat ist seit langem bekannt, ein Beweis dazu befindet sich in [Mal04].

Lemma 2.7 *Jede Turingmaschine M kann effektiv in eine Turingmaschine M' umgewandelt werden, die folgende Eigenschaften hat:*

1. *M' schreibt nie das Blanksymbol.*
2. *In jeder Berechnung von M werden mindestens zwei Schritte ausgeführt.*
3. *M' akzeptiert nie nach einer ungeraden Anzahl von Berechnungsschritten.*

Definition 2.26 *Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ eine Turingmaschine und $\# \notin Q \cup \Gamma$. Wir können außerdem ohne Beschränkung der Allgemeinheit annehmen, daß $Q \cap \Gamma = \emptyset$ gilt. Eine akzeptierende Berechnung von M ist eine Zeichenkette*

$$ID_0 \# ID_1^R \# ID_2 \# ID_3^R \# \dots \# ID_{2k} \# (ID_{2k+1})^R,$$

so daß

1. *jede ID_i eine Zustandsbeschreibung (ID) von M, also ein String in $\Gamma^* Q \Gamma^*$ ist, der nicht mit B endet,*
2. *ID_0 eine Anfangs-ID, also in der Form $q_0 w$ für $w \in \Sigma^*$ ist,*
3. *ID_{2k+1} eine akzeptierende ID, also in der Form $\Gamma^* F \Gamma^*$ ist,*
4. *$ID_i \vdash_M ID_{i+1}$ gilt für $1 \leq i \leq 2k$.*

Wir bezeichnen die akzeptierenden Berechnungen oder „valid computations“ einer Turingmaschine M mit *VALC*(M). Das Komplement dieser Sprache, das genau aus den Wörtern besteht, die keine akzeptierende Berechnung darstellen, ist die Sprache der „invalid computations“. Formal schreiben wir

$$\mathit{INVALC}(M) = (Q \cup \Gamma \cup \{\#\})^* \setminus \mathit{VALC}(M).$$

Schon in [Har67] wurde gezeigt, daß *INVALC*(M) eine kontextfreie Sprache ist. Für die folgenden Kapitel brauchen wir ein etwas schärferes Ergebnis.

Lemma 2.8 *Für jede Turingmaschine M ist INVALC$(M) \in \mathcal{L}$(LIN).*

Beweis: Die Details des Beweises finden sich in [Mal04]. Wenn ein Wort w in *INVALC*(M) enthalten ist, dann trifft einer der folgenden Punkte zu:

1. w hat nicht die Form $ID_0 \# ID_1^R \# \ldots \# ID_{2n} \# ID_{2n+1}^R \#$, wobei ID_i eine ID von M ist.
2. ID_0 ist keine Anfangskonfiguration. Das heißt, daß $ID_0 \notin q_0 \Sigma^*$ ist.
3. ID_{2n+1} ist keine akzeptierende Konfiguration. Das heißt, daß $ID_{2n+1} \notin \Gamma^* F \Gamma^*$ ist.
4. $ID_i \vdash_M ID_{i+1}$ trifft nicht zu für ein gerades i mit $0 \leq i \leq 2n$.
5. $ID_i \vdash_M ID_{i+1}$ trifft nicht zu für ein ungerades i mit $1 \leq i \leq 2n+1$.

Die Sprachen, die durch die Punkte (1), (2) und (3) beschrieben werden, sind reguläre Mengen.

Satz 2.3 sagt aus, daß die linearen Sprachen gerade die Klasse von Sprachen sind, die von 1-turn-PDAs akzeptiert werden. Ein 1-turn-PDA A, der die Sprache, die durch Punkt (4) beschrieben wird, akzeptiert, arbeitet wie folgt. Die Eingabe wird gelesen und dabei rät A, welcher Übergang $ID_i \vdash_M ID_{i+1}$ für ein gerades i falsch ist. Ein gerades i erkennt A an einer geraden Anzahl von vorangegangenen Symbolen $\#$. Während ID_i gelesen wird, speichert A einen String α auf dem Keller, für den es eine Transition $ID_i \vdash_M \alpha$ gibt. Das erste Symbol von α ist dabei am weitesten unten auf dem Keller. Nachdem das nächste Symbol $\#$ gelesen wurde, vergleicht A die weiteren Symbole der Eingabe (ID_{i+1}) mit den Symbolen auf dem Keller. In der Definition aller gültigen $w = ID_0 \# ID_1^R \# \ldots \# ID_{2n} \# ID_{2n+1}^R \#$ sehen wir, daß ID_{i+1} rückwärts aufgeführt ist. Daher ist dieser Vergleich möglich. Wenn $\alpha \neq ID_{i+1}$, liest A die übrige Eingabe und akzeptiert. Es sei weiterhin bemerkt, daß eine Turingmaschine M in einem Schritt ihren Schreib-/ Lesekopf höchstens eine Bandzelle

nach links oder rechts bewegt und höchstens den Inhalt einer Bandzelle ändert. Also kann A die Schritte von M auf ID_i durch seine endliche Kontrolle simulieren und das Resultat auf den Keller schreiben. Die Kellerbewegungen von A bestehen aus einer Phase, in der nur Symbole geschrieben werden und einer Phase, in der nur Symbole gelesen und entfernt werden. A ist also ein 1-turn-PDA.

Auf ähnliche Weise kann man einen PDA A' konstruieren, der die Sprache, die durch Punkt (5) beschrieben wird, akzeptiert. Auch A' ist ein 1-turn-PDA.

Insgesamt ist also $\mathit{INVALC}(M)$ die Vereinigung aus fünf linearen Sprachen. Die linearen Sprachen sind unter Vereinigung abgeschlossen (siehe nächster Abschnitt), daher ist auch $\mathit{INVALC}(M)$ eine lineare Sprache. □

2.2.3 Abschlußeigenschaften und Entscheidbarkeit

Bei der genaueren Untersuchung von Klassen formaler Sprachen sind der Abschluß dieser Klassen unter bestimmten Operationen oder die Tatsache, daß eine Klasse gerade nicht unter einer Operation abgeschlossen ist, ein Hilfsmittel für Konstruktionen und Beweisführung.

Außerdem ist es von Bedeutung, ob bestimmte Fragestellungen für bestimmte Klassen von formalen Sprachen beantwortet werden können. Unter diese Fragestellungen fällt zum Beispiel die Frage nach Leere, Endlichkeit und Unendlichkeit einer Sprache, aber auch die Frage, ob eine Sprache aus einer Klasse $\mathcal{L}_1$ auch in einer anderen Klasse $\mathcal{L}_2$ enthalten ist.

Wir führen als erstes die Operationen ein, die wir in den folgenden Kapiteln benutzen werden.

Definition 2.27 *Eine Sprachklasse $\mathcal{L}$ ist **abgeschlossen** unter*

1. ***Vereinigung**, wenn für $L_1, L_2 \in \mathcal{L}$ auch*

   $$L_1 \cup L_2 = \{w \mid w \in L_1 \text{ oder } w \in L_2\} \in \mathcal{L} \text{ gilt.}$$

2. ***Schnitt**, wenn für $L_1, L_2 \in \mathcal{L}$ auch*

   $$L_1 \cap L_2 = \{w \mid w \in L_1 \text{ und } w \in L_2\} \in \mathcal{L} \text{ gilt.}$$

3. ***Komplement**, wenn für $L_1 \in \mathcal{L}$, $alph(L_1) = \Sigma$ auch*

   $$\overline{L_1} = \Sigma^* \setminus L_1 \in \mathcal{L} \text{ gilt.}$$

4. ***Homomorphismus**, wenn für $L_1 \in \mathcal{L}$, $alph(L_1) = \Sigma_1$ und einen Homomorphismus $h : \Sigma_1^* \to \Sigma_2^*$ auch*

   $$h(L_1) = \{h(w) \mid w \in L_1\} \in \mathcal{L} \text{ gilt.}$$

5. ***inversem Homomorphismus**, wenn für $L_1 \in \mathcal{L}$, $alph(L_1) = \Sigma_1$ und einen Homomorphismus $h : \Sigma_2^* \to \Sigma_1^*$ auch*

   $h^{-1}(L_1) = \{w \mid h(w) \in L_1\} \in \mathcal{L}$ *gilt.*

6. ***Schnitt mit regulären Mengen**, wenn für $L_1 \in \mathcal{L}$ und $R \in REG$ auch*

   $L \cap R \in \mathcal{L}$ *gilt.*

7. ***Konkatenation**, wenn für $L_1, L_2 \in \mathcal{L}$ auch*

   $L_1 \cdot L_2 \in \mathcal{L}$ *gilt.*

8. ***Kleenesche Hülle**, wenn für $L_1 \in \mathcal{L}$ auch*

   $L_1^* \in \mathcal{L}$ *gilt.*

Eine Sprachklasse, die unter Homomorphismus, inversem Homomorphismus und Schnitt mit regulären Mengen abgeschlossen ist, wird auch ***volles Trio*** oder ***semi-AFL*** genannt. Falls sie zusätzlich unter Vereinigung, Konkatenation und Kleenescher Hülle abgeschlossen ist, heißt sie ***volle AFL***.

In Tabelle 2.1 sind die Abschlußeigenschaften einiger Sprachklassen zusammengefaßt. Ein $\surd$ in der entsprechenden Zelle bedeutet, daß die Sprachklasse unter der Operation abgeschlossen ist, und eine leere Zelle bedeutet, daß die Sprachklasse unter der Operation nicht abgeschlossen ist. Die De Morgan-

Abschlußeigenschaft	$\mathcal{L}(REG)$	$\mathcal{L}(LIN)$	$\mathcal{L}(mLIN)$	$\mathcal{L}(CF)$	$\mathcal{L}(RE)$
Vereinigung	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Schnitt	$\surd$				$\surd$
Komplement	$\surd$				
Homomorphismus	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
inverser Homomorphismus	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Schnitt mit $R \in \mathcal{L}(REG)$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Konkatenation	$\surd$			$\surd$	$\surd$
Kleenesche Hülle	$\surd$			$\surd$	$\surd$

Tabelle 2.1: Abschlußeigenschaften

schen Regeln stellen den Zusammenhang zwischen Schnitt, Komplement und Vereinigung dar.

Satz 2.4 (De Morgansche Regeln) *Seien V und W beliebige Mengen, dann gilt*

1. $\overline{V \cap W} = \overline{V} \cup \overline{W}$,
2. $\overline{V \cup W} = \overline{V} \cap \overline{W}$.

Eine weitere Operation, die im Kapitel 6 benutzt wird, ist die gsm-Abbildung.

Definition 2.28 *Eine **gsm** (kurz für „generalized sequential machine") ist ein 6-Tupel $M = (Q, \Sigma, \Delta, \delta, q_0, F)$, wobei*

1. *Q eine endliche Menge von Zuständen,*
2. *Σ das Eingabealphabet,*
3. *Δ das Ausgabealphabet,*
4. *δ die Übergangsrelation von der Menge $Q \times \Sigma$ in die endliche Teilmengen von $Q \times \Delta^*$,*
5. *$q_0 \in Q$ der Anfangszustand und*
6. *$F \subseteq Q$ die Menge der Endzustände ist.*

Die Übergangsrelation wird wie üblich auf $Q \times \Sigma^*$ wie folgt erweitert:

$$\delta(q, \varepsilon) = \delta(q, \varepsilon)$$

und

$$\delta(q, xa) = \{(p, w_1 w_2) \mid (p', w_1) \in \delta(q, x), (p, w_2) \in \delta(p', a), p' \in Q\}$$

für $x \in \Sigma^*$ und $a \in \Sigma$.

Sei M eine gsm, dann definieren wir $M(x) = \{y \mid (p, y) \in \delta(q_0, x)\}$. Außerdem ist für eine Sprache L die Menge $M(L)$ definiert durch $M(L) = \{y \mid y \in M(x)$ für ein $x \in L\}$. Wir nennen $M(L)$ auch gsm-Abbildung.

Lemma 2.9 *[HU79] Jede Sprachklasse, die unter Homomorphismus, inversem Homomorphismus und Schnitt mit regulären Mengen abgeschlossen ist, ist auch unter gsm-Abbildungen abgeschlossen.*

Ein Problem oder eine Fragestellung wird ***entscheidbar*** genannt, wenn es einen Algorithmus gibt, der als Eingabe eine Instanz des Problems hat und entscheidet, ob die Antwort „ja" oder „nein" ist.

Der Begriff der Entscheidbarkeit hängt eng mit dem Modell der Turingmaschine zusammen, da die Tatsache, ob die Eigenschaft einer Sprache algorithmisch bestimmt werden kann, gleichbedeutend damit ist, ob sie von einer Turingmaschine errechnet werden kann, die auf jeder Eingabe hält. Ein Problem ist also entscheidbar, wenn seine Sprache rekursiv ist.

In Tabelle 2.2 sind die Entscheidbarkeitsfragen für einige der oben eingeführter Sprachklassen aufgelistet. Dabei sind L, L_1 und L_2 in der der Spalte entsprechenden Sprachklasse. Außerdem gilt $L, L_1, L_2 \subseteq \Sigma^*$ und $w \in \Sigma^*$. Ein $\surd$ in der entsprechenden Zelle bedeutet dabei, daß die Fragestellung für die Sprachklasse entscheidbar ist. Eine leere Zelle bedeutet, daß die Fragestellung für die Sprachklasse nicht entscheidbar ist.

Der Satz von Rice ist eines der wichtigsten Ergebnisse über die Entscheidbarkeit bei rekursiv aufzählbaren Mengen. Bevor wir ihn hier aufführen, müssen einige Begriffe erklärt werden.

Fragestellung	$\mathcal{L}(REG)$	$\mathcal{L}(CF)$	$\mathcal{L}(RE)$
Ist $w \in L$?	$\surd$	$\surd$	
Ist $L = \emptyset$?	$\surd$	$\surd$	
Ist $L = \Sigma^*$?	$\surd$		
Ist $L_1 = L_2$?	$\surd$		

Tabelle 2.2: Entscheidbarkeitseigenschaften

Sei M eine beliebige Turingmaschine. Wir können eine Beschreibung dieser Turingmaschine über dem Alphabet {0,1} kodieren. Ein Verfahren zur Kodierung ist zum Beispiel in [HU79] beschrieben. Mit $\langle M \rangle$ bezeichnen wir die ***Kodierung*** der Turingmaschine M.

Sei $\mathcal{P}$ eine Menge von rekursiv aufzählbaren Mengen, die Teilmengen von $\{0,1\}^*$ sind. $\mathcal{P}$ wird auch ***Eigenschaft*** der rekursiv aufzählbaren Sprachen genannt. Eine Menge L hat die Eigenschaft $\mathcal{P}$, wenn L ein Element von $\mathcal{P}$ ist.

Als Beispiel betrachten wir die Eigenschaft Unendlichkeit, die als Menge $\{L \mid L$ ist unendlich $\}$ dargestellt wird. $\mathcal{P}$ ist eine ***triviale*** Eigenschaft, wenn $\mathcal{P}$ aus allen rekursiv aufzählbaren Sprachen besteht oder leer ist.

Weiterhin ist die Menge $L_{\mathcal{P}}$ definiert mit

$$L_{\mathcal{P}} = \{\langle M \rangle \mid T(M) \in \mathcal{P}\},$$

wobei M eine Turingmaschine ist.

Satz 2.5 (Satz von Rice) *Jede nicht-triviale Eigenschaft $\mathcal{P}$ einer rekursiv aufzählbaren Sprache ist unentscheidbar.*

Der nächste Satz beinhaltet ein ähnliches Ergebnis über $L_{\mathcal{P}}$. Wir werden ihn für Ergebnisse in der Beschreibungskomplexität benutzen.

Satz 2.6 (Satz von Rice für rekursiv aufzählbare Indexmengen) *Sei $\mathcal{P}$ eine Menge rekursiv aufzählbarer Sprachen und $L_{\mathcal{P}}$ definiert wie oben. $L_{\mathcal{P}}$ ist rekursiv aufzählbar genau dann, wenn folgende Punkte zutreffen.*

1. *Wenn $L \in \mathcal{P}$ und $L \subseteq L'$ für eine rekursiv aufzählbare Menge L', dann ist $L' \in \mathcal{P}$.*
2. *Wenn L eine unendliche Menge ist mit $L \in \mathcal{P}$, dann gibt es eine endliche Teilmenge T von L mit $T \in \mathcal{P}$.*
3. *Die Menge der endlichen Mengen in $\mathcal{P}$ ist abzählbar.*

2.3 Beschreibungskomplexität

Klassische Fragestellungen im Bereich der formalen Sprachen beschäftigen sich mit der generativen Mächtigkeit von Beschreibungssystemen wie zum

Beispiel Klassen von Automaten oder Grammatiken. Außerdem werden Abschlußeigenschaften und die Entscheidbarkeit von Fragestellungen bei den entsprechenden Sprachklassen untersucht.

In der Beschreibungskomplexität hingegen steht die Größe oder Komplexität eines Beschreibungssystems im Vordergrund. Beispielsweise sucht man nach der minimalen Beschreibung für eine bestimmte Sprache innerhalb einer bestimmten Klasse. Außerdem beschäftigt man sich mit sogenannte Tradeoffs. Gegeben ist dabei eine Beschreibung in einem ersten System. Dann wird untersucht, wie groß eine äquivalente Beschreibung in einem zweiten System mindestens oder höchstens sein muß.

Wir folgen [GKK+02] bei den grundlegenden Definitionen.

Definition 2.29 *Ein Beschreibungssystem $\mathcal{K}$ für eine Klasse $\mathcal{L}$ von Sprachen ist eine Menge von endlichen Beschreibungen, die jeweils eine Sprache in $\mathcal{L}$ darstellen. Für jede Sprache $L \in \mathcal{L}$ sei*

$$\mathcal{K}(L) = \{K \in \mathcal{K} \text{ und } K \text{ stellt } L \text{ dar}\}.$$

Definition 2.30 *Ein Komplexitätsmaß $\mathcal{C}$ für ein Beschreibungssystem $\mathcal{K}$ ist eine rekursive, totale Funktion*

$$\mathcal{C} : \mathcal{K} \to \mathbb{R}_0^+.$$

Dabei ist $\mathcal{C}(K)$ die Komplexität einer Beschreibung $K \in \mathcal{K}$.

Die oben kurz angeführten Fragestellungen in der Beschreibungskomplexität können jetzt formal dargestellt werden.

Seien $\mathcal{K}_1$ und $\mathcal{K}_2$ zwei Beschreibungssysteme und $\mathcal{L} = \mathcal{L}(\mathcal{K}_1) \cap \mathcal{L}(\mathcal{K}_2)$ unendlich. Außerdem seien $\mathcal{C}_1 : \mathcal{K}_1 \to \mathbb{R}_0^+$ und $\mathcal{C}_2 : \mathcal{K}_2 \to \mathbb{R}_0^+$ Komplexitätsmaße.

- Gibt es eine Funktion $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$, so daß für alle $L \in \mathcal{L}$

  $$\min\{\mathcal{C}_2(K) \mid K \in \mathcal{K}_2(L)\} \leq f(\min\{\mathcal{C}_1(K') \mid K' \in \mathcal{K}_1(L)\}?$$

  Die Änderung der Größe einer Beschreibung, wenn man vom Beschreibungssystem $\mathcal{K}_1$ zum Beschreibungssystem $\mathcal{K}_2$ wechselt, nennt man den ***Tradeoff***. Wenn f existiert, was nicht unbedingt der Fall sein muß, ist f eine ***obere Schranke*** für den Tradeoff.

- Gibt es eine unendliche Folge von Sprachen $(L_i)_{i=0}^{\infty}$ mit $L_i \in \mathcal{L}$ für $i \geq 0$ und eine Funktion $f' : \mathbb{R}_0^+ \to \mathbb{R}_0^+$, so daß für alle $i \geq 0$

  $$\min\{\mathcal{C}_2(K) \mid K \in \mathcal{K}_2(L_i)\} \geq f'(\min\{\mathcal{C}_1(K') \mid K' \in \mathcal{K}_1(L_i)\}?$$

  Wenn diese Funktion f' existiert, ist sie eine ***untere Schranke*** für den Tradeoff beim Wechsel des Beschreibungssystems von $\mathcal{K}_1$ zu $\mathcal{K}_2$.

Wenn wir jetzt Ergebnisse über bestimmte Maße zeigen wollen, werden wir häufig keine Schranken wie eben definiert finden können. In manchen Fällen ist es nicht möglich, eine Funktion oder auch nur Größenordnung für den

Tradeoff anzugeben, weil es eine solche Funktion nicht gibt. Es ist aber auch möglich, daß Hilfsmittel für die genaue Bestimmung des Tradeoffs fehlen und deshalb keine Funktion angegeben werden kann.

Bei der Untersuchung von Grammatiken und Grammatiksystemen wurden für den Fall, daß die obere Schranke nicht genau bestimmt werden kann oder keine obere Schranke für den Tradeoff zwischen zwei Beschreibungssystemen existiert, eine Reihe von Möglichkeiten definiert, diesen Tradeoff zu klassifizieren. Wir folgen bei der Definition [DP91].

Definition 2.31 *Sei $\mathcal{C}$ ein Maß für die Komplexität einer Klasse von Beschreibungen und $\mathcal{K}$ ein Beschreibungssystem. Für eine Sprache $L \in \mathcal{L}(\mathcal{K})$ definieren wir*

$$\mathcal{C}_{\mathcal{K}}(L) = \min\{\mathcal{C}(K) \mid K \in \mathcal{K}(L)\}.$$

Definition 2.32 *Seien $\mathcal{K}_1$ und $\mathcal{K}_2$ zwei Beschreibungssysteme und sei weiterhin $\mathcal{L} = \mathcal{L}(\mathcal{K}_1) \cap \mathcal{L}(\mathcal{K}_2)$ unendlich. Sei $\mathcal{C}_1$ ein Maß für $\mathcal{K}_1$ und $\mathcal{C}_2$ eine Maß für $\mathcal{K}_2$. Außerdem sei*

$$\mathcal{C}_{1\mathcal{K}_1}(L) \leq \mathcal{C}_{2\mathcal{K}_2}(L) \text{ für alle } L \in \mathcal{L}.$$

Wir setzen

1. $\mathcal{K}_1(\mathcal{C}_1) = \mathcal{K}_2(\mathcal{C}_2)$ *genau dann, wenn*

   $$\mathcal{C}_{1\mathcal{K}_1}(L) = \mathcal{C}_{2\mathcal{K}_2}(L) \text{ ist für alle } L \in \mathcal{L}.$$

2. $\mathcal{K}_1(\mathcal{C}_1) <_1 \mathcal{K}_2(\mathcal{C}_2)$ *genau dann, wenn eine Sprache $L \in \mathcal{L}$ existiert mit*

   $$\mathcal{C}_{1\mathcal{K}_1}(L) < \mathcal{C}_{2\mathcal{K}_2}(L).$$

3. $\mathcal{K}_1(\mathcal{C}_1) <_2 \mathcal{K}_2(\mathcal{C}_2)$ *genau dann, wenn eine Folge von Sprachen $L_n \in \mathcal{L}$, $n \geq 1$ existiert mit*

   $$\mathcal{C}_{2\mathcal{K}_2}(L_n) - \mathcal{C}_{1\mathcal{K}_1}(L_n) > n.$$

4. $\mathcal{K}_1(\mathcal{C}_1) <_3 \mathcal{K}_2(\mathcal{C}_2)$ *genau dann, wenn eine Folge von Sprachen $L_n \in \mathcal{L}$, $n \geq 1$ existiert mit*

   $$\lim_{n\to\infty} \frac{\mathcal{C}_{1\mathcal{K}_1}(L_n)}{\mathcal{C}_{2\mathcal{K}_2}(L_n)} = 0.$$

5. $\mathcal{K}_1(\mathcal{C}_1) <_4 \mathcal{K}_2(\mathcal{C}_2)$ *genau dann, wenn eine Folge von Sprachen $L_n \in \mathcal{L}$, $n \geq 1$ existiert mit $p \geq 1$ und*

   $$\mathcal{C}_{2\mathcal{K}_2}(L_n) > n, \mathcal{C}_{2\mathcal{K}_1}(L_n) \leq p.$$

Die eben aufgeführten Klassifikationen von Tradeoffs sind nur dann aussagekräftig, wenn man für beide Beschreibungssysteme gleiche oder ähnlich geartete Maße benutzt, da man mit einem schwachen Maß für $\mathcal{K}_1$ und einem sehr starken Maß für $\mathcal{K}_2$ leicht Ergebnisse im Bereich $<_3$ und $<_4$ erreichen kann, die aber wegen der Unterschiedlichkeit der Maße nicht sehr aussagekräftig sind.

Für die nichtrekursiven Tradeoffs im letzten Abschnitt dieses Kapitels werden wir daher auch noch stärkere Einschränkungen für die Eigenschaften von Maßen für Beschreibungssysteme machen.

2.4 Maße für die Größe von Grammatiken und Grammatiksystemen

Wenn man die Komplexität oder Größe einer Beschreibung betrachtet, braucht man auch Maße, mit denen man diese Größe bestimmen kann. In diesem Abschnitt werden wir uns mit Maßen für die Größe von Grammatiken und auch Grammatiksystemen beschäftigen.

Eine genaue Definition von Grammatiksystemen befindet sich im Kapitel 4. Für diesen Abschnitt ist es ausreichend, im Gegensatz zur kontextfreien Grammatik mit einer Menge von Produktionen

$$G = (N, T, P, S)$$

ein kontextfreies CD Grammatiksystem Γ als Grammatik mit n Mengen von kontextfreien Produktionen zu definieren, die sequentiell an einer gemeinsamen Satzform arbeiten

$$\Gamma = (N, T, P_1, \ldots, P_n, S).$$

Die einzelnen Produktionsmengen werden auch als Komponenten bezeichnet.

Wenn man die Beschreibungskomplexität von kontextfreien Grammatiken betrachtet, ist als Maß zuerst einmal die Länge der Beschreibung naheliegend. Allerdings können auch verschiedene andere Parameter für die Komplexität einer Beschreibung durch eine Grammatik wichtig sein. Im Gegensatz zur reinen Größe wird dabei mehr Wert auf Strukturmerkmale Strukturmerkmale gelegt. Oft betrachtet man in diesem Zusammenhang die Anzahl der Produktionen und die Anzahl der Nichtterminale. Aus [Gru67] und [Gru69] sind folgende Maße für kontextfreie Grammatiken bekannt:

$$\begin{aligned} Var(G) &= |N|, \\ mProd(G) &= |P|, \\ Symb(G) &= \sum_{A \to \alpha \in P} (|\alpha| + 2). \end{aligned}$$

Für CD Grammatiksysteme Γ stellen sich erst einmal ähnliche Fragen wie für kontextfreie Grammatiken. Man kann dafür die oben eingeführten Maße ähnlich oder sogar gleich definieren, wir folgen dabei [BR02] und [DP91]:

$$\begin{aligned} Var(\Gamma) &= |N|, \\ mProd(\Gamma) &= \sum_{i=1}^{n} |P_i|, \\ Symb(\Gamma) &= \sum_{i=1}^{n} \sum_{A \to \alpha \in P_i} (|\alpha| + 2). \end{aligned}$$

Allerdings ist auch die Anzahl der Komponenten in einem CD Grammatiksystem ein Parameter, mit dessen Größe die Komplexität der Beschreibung steigt. Wir definieren ihn mit

$$Deg(\Gamma) = n.$$

Es stellt sich nicht nur die Frage, wieviele Produktionen man insgesamt zur Beschreibung einer bestimmten Sprachklasse braucht, sondern auch wieviele Produktionen man mindestens in einer Komponente zur Beschreibung einer bestimmten Sprachklasse benötigt. Dieses Maß heißt im folgenden

$$cProd(\Gamma) = \max\{|P_i| \mid 1 \leq i \leq n\}.$$

Um die Größe der einzelnen Komponenten zusammen mit der Anzahl der Komponenten in einem Maß betrachten zu können definieren wir das Produkt aus beiden Parametern.

$$cVar(\Gamma) = n \cdot |N|.$$

Anhand eines Beispiels werden wir jetzt die Begriffe, die zur Beschreibungskomplexität von Grammatiken und CD Grammatiksystemen eingeführt wurden, näher erläutern.

Beispiel 2.1 In unserem Beispiel ist das betrachtete Beschreibungssystem *CF*, die Klasse der kontextfreien Grammatiken, und das betrachtete Maß $mProd$, die Anzahl der Produktionen. Wir betrachten für $n \geq 1$ die Folge von Sprachen:

$$L_n = \{a^{2i} \mid 0 \leq i \leq n-1\}.$$

Wir können jetzt für jede Sprache L_n eine kontextfreie Grammatik G_n angeben mit $L(G_n) = L_n$:

$$G_n = (\{S\}, \{a\}, \{S \to a^{2i} \mid 0 \leq i \leq n-1\}, S).$$

Man kann sehen, daß $mProd(G_n) = n$. Durch diese Folge von Grammatiken haben wir jetzt eine Beschränkung für $mProd_{CF}(L_n)$ gefunden, denn es gilt

$mProd_{CF}(L_n) \leq n$. Es bleibt allerdings die Frage, ob es auch Grammatiken mit weniger Produktionen gibt, die die Sprachen L_n erzeugen.

Für $n = 1$ ist das sicher nicht der Fall. Auch im Fall $n = 2$ kann man mit einfachen Argumenten zeigen, daß die oben angegebene Grammatik von der Zahl der Produktionen her optimal ist. Angenommen wir hätten eine Grammatik G', die L_2 mit nur einer Produktion erzeugt. G' muß auf jeden Fall eine Produktion enthalten, auf deren rechter Seite ε steht, sonst kann ε nicht erzeugt werden. G' muß aber auch eine Produktion enthalten, auf deren rechter Seite ein Terminal a enthalten ist, was der Annahme widerspricht. Es gilt also

$$mProd_{CF}(L_2) = 2.$$

In [Gru69] wird weiterhin gezeigt, daß jede kontextfreie Grammatik für L_n mindestens n Produktionen enthalten muß. Es gilt also auch

$$mProd_{CF}(L_n) = n.$$

Im nächsten Beispiel wollen wir das eben gewonnene Ergebnis über das Maß $mProd$ bei kontextfreien Grammatiken mit Ergebnissen für dieses Maß bei CD Grammatiksystemen in Relation setzten, um die Vergleiche zwischen zwei Beschreibungssystemen bezüglich eines Maßes zu verdeutlichen.

Beispiel 2.2 Seien $a_1, a_2, \ldots a_n$ paarweise verschiedene Terminale und sei für $1 \leq i \leq r$

$$L_{n,r} = \{a_r^{2i} \mid 1 \leq i \leq r\} \quad \text{und} \quad L_n = \bigcup_{r=1}^{n} L_{n,r}.$$

Wir haben im Beispiel 2.1 gesehen, daß $mProd_{CF}(L_{n,r}) = n$ und daher ist

$$mProd_{CF}(L_n) = n^2.$$

Wir geben jetzt ein CD Grammatiksystem an, das L_n erzeugt. Das CD Grammatiksystem arbeitet im t-Modus. Das heißt, daß jede Ableitung mit dem Startsymbol anfängt und jeweils eine Komponente an der Satzform arbeitet. Jede Komponente muß so lange Produktionen anwenden, bis sie keine weiteren Produktionen anwenden kann. Eine genaue Erklärung der Funktionsweise dieses CD Grammatiksystems ist in Beispiel 4.2 zu finden.

$$\begin{aligned}
\Gamma_n &= (\{S, A\}, \{a_1, \ldots, a_n\}, P_1, \ldots, P_{n+1}, S), \\
P_1 &= \{A \to a_1\}, \\
&\ldots \\
P_n &= \{A \to a_n\}, \\
P_{n+1} &= \{S \to A^{2i} \mid 1 \leq i \leq n\}.
\end{aligned}$$

Wie man sehen kann, ist $mProd(\Gamma_n) = 2n$, also $mProd_{(CD\text{-}CF,t)}(L_n) \leq 2n$. Wenn man jetzt ausrechnet, wie das Verhältnis zwischen $mProd_{(CD\text{-}CF,t)}(L_n)$

und $mProd_{CF}(L_n)$ ist, erhält man

$$\lim_{n\to\infty} \frac{mProd_{(CD\text{-}CF,t)}(L_n)}{mProd_{CF}(L_n)} \leq \lim_{n\to\infty} \frac{2n}{n^2} = 0.$$

Nach den Definition aus dem letzten Abschnitt ist dann also

$$(CD\text{-}CF, t)(mProd) <_3 CF(mProd).$$

2.5 Nichtrekursive Tradeoffs

In diesem Abschnitt beschäftigen wir uns auch mit der Möglichkeit, Aussagen über die Veränderung der Größe der Beschreibung einer Sprache zu machen, wenn man von einem Beschreibungssystem zu einem anderen wechselt. Dabei beziehen wir uns aber nicht auf bestimmte Maße, sondern auf alle Maße, die bestimmten Voraussetzungen entsprechen.

Außerdem wird die Vergrößerung der Beschreibung (oder der Tradeoff) beim Wechsel von einem Beschreibungssystem zu einem anderen nicht durch eine rekursive Funktion beschränkt sein. Wir werden also nichtrekursiven Tradeoffs erhalten.

Es sei noch angemerkt, daß der nichtrekursive Tradeoff sehr ähnlich zur Definition $<_4$ aus dem letzten Abschnitt ist. Im Fall $<_4$ wird jedoch im Bezug auf ein bestimmtes Größenmaß verglichen, und die Vergrößerung beim Wechsel von der stärkeren zur schwächeren Beschreibung ist nicht nur durch keine rekursive, sondern durch überhaupt keine Funktion begrenzt.

Zuerst werden wir einige Begriffe als Vorbereitung zur Definition der nichtrekursiven Tradeoffs einführen.

Definition 2.33 *Seien $\mathcal{K}_1$ und $\mathcal{K}_2$ zwei Klassen von Beschreibungssystemen und $\mathcal{L}(\mathcal{K}_1)$ und $\mathcal{L}(\mathcal{K}_2)$ die entsprechenden Sprachklassen. Sei weiterhin $\mathcal{L} = \mathcal{L}(\mathcal{K}_1) \cap \mathcal{L}(\mathcal{K}_2)$ unendlich. Als nächstes seien $\mathcal{C}_1 : \mathcal{K}_1 \to \mathbb{N}$ und $\mathcal{C}_2 : \mathcal{K}_2 \to \mathbb{N}$ zwei Komplexitätsmaße für Systeme aus $\mathcal{K}_1$ bzw. $\mathcal{K}_2$. Das Urbild von $\mathcal{C}_1$ und $\mathcal{C}_2$ für jede Zahl $n \in \mathbb{N}$ ist endlich.*

Wenn es jetzt zu jeder rekursiven Funktion f eine Folge von Sprachen $L_n \in \mathcal{L}$, $n \geq 1$ gibt, für die gilt:

1. *für alle i existiert ein $M_i \in \mathcal{K}_1$ mit $L(M_i) = L_i$ und*

   $$\mathcal{C}_1(M_{i+1}) > \mathcal{C}_1(M_i) \text{ für alle } i \geq 1,$$

2. *für alle i und alle $M \in \mathcal{K}_2$ mit*

   $$L(M) = L_i, \quad \mathcal{C}_2(M) > f(\mathcal{C}_1(M_i)),$$

*dann besteht zwischen $\mathcal{K}_1$ und $\mathcal{K}_2$ ein **nichtrekursiver Tradeoff**.*

Für einen nichtrekursiven Tradeoff zwischen $\mathcal{K}_1$ und $\mathcal{K}_2$ schreiben wir auch

$$\mathcal{K}_2 \overset{nonrec}{\longleftarrow} \mathcal{K}_1.$$

Für den Nachweis von nichtrekursiven Tradeoffs verwenden wir eine Methode von Hartmanis aus [Har79] und [Har80]. Um die Methode zu verwenden, werden die „invalid computations" einer Turingmaschine $\mathit{INVALC}(M)$ und deren Eigenschaften benutzt.

Die von Hartmanis eingeführte Methode wird im folgenden in einer Form benutzt, die etwas allgemeiner als das Original formuliert ist.

Satz 2.7 *[Mal02] Seien $\mathcal{K}_1$ und $\mathcal{K}_2$ zwei Beschreibungssysteme für rekursive Sprachen. Wenn für jede Turingmaschine M eine Sprache $L_M \in \mathcal{L}(\mathcal{K}_1)$ und deren Deskriptor effektiv konstruiert werden können, so daß genau dann $L_M \in \mathcal{L}(\mathcal{K}_2)$ ist, wenn $T(M)$ endlich ist, dann ist der Tradeoff zwischen $\mathcal{K}_1$ und $\mathcal{K}_2$ nichtrekursiv.*

Beweis: Um die Vorgehensweise verständlicher zu machen, wird hier der Beweis angegeben. Der Beweis wird durch Widerspruch geführt. Wir nehmen also zuerst an, daß der Tradeoff zwischen $\mathcal{K}_1$ und $\mathcal{K}_2$ rekursiv ist und deshalb durch eine rekursive Funktion f nach oben beschränkt werden kann. Dann gibt es für jedes $K \in \mathcal{K}_1$ mit $L(K) \in \mathcal{L}(\mathcal{K}_1) \cap \mathcal{L}(\mathcal{K}_2)$ ein $K' \in \mathcal{K}_2$ mit $L(K) = L(K')$ und $\mathcal{C}_2(K') \leq f(\mathcal{C}_1(K))$.

Sei K eine beliebige Beschreibung in $\mathcal{K}_1$. Wir betrachten jetzt alle Beschreibungen $K_1, K_2, \ldots, K_m \in \mathcal{K}_2$, für die gilt $\mathcal{C}_2(K_i) \leq f(\mathcal{C}_1(K))$, $1 \leq i \leq m$. Die Menge dieser Beschreibungen ist endlich, da das Urbild von $\mathcal{C}_1$ und $\mathcal{C}_2$ endlich ist.

Es ist $L(K)$ genau dann nicht in $\mathcal{L}(\mathcal{K}_2)$, wenn $L(K_i) \neq L(K)$ für alle $1 \leq i \leq m$, sonst wäre f keine obere Schranke für den Tradeoff. $L(K_i) \neq L(K)$ gilt wiederum genau dann, wenn es für jedes K_i mit $1 \leq i \leq m$ ein w_i gibt mit $w_i \in L(K) \setminus L(K_i)$ oder $w_i \in L(K_i) \setminus L(K)$.

Man kann jetzt eine Turingmaschine M' konstruieren, die für die Eingabe $\langle K \rangle$, also die Kodierung von K folgendes leistet:

1. M' generiert alle Beschreibungen $K_1, \ldots, K_m$, $1 \leq i \leq m$.
2. M' setzt den Zähler j auf 0.
3. M' erzeugt alle Eingabewörter w mit $|w| = j$.
4. M' prüft, ob $w \in L(K) \setminus L(K_i)$ oder $w \in L(K_i) \setminus L(K)$ für alle $1 \leq i \leq m$.
5. Wenn für alle i ein w_i gefunden wurde, das in $L(K) \setminus L(K_i)$ oder in $L(K_i) \setminus L(K)$ ist, dann hält M' an und akzeptiert.
6. Wenn nicht, dann fährt M' fort bei Punkt 3. für $j = j + 1$.

Die Turingmaschine M' hält für alle Eingaben $\langle K \rangle$ mit $K \in \mathcal{K}_1 \setminus \mathcal{K}_2$ an und akzeptiert. Bei den anderen Eingaben hält sie nicht an. Es ist also

$$T(M') = \{\langle K \rangle \mid K \in \mathcal{K}_1 \setminus \mathcal{K}_2\}$$

und damit die Menge $\{\langle K\rangle \mid K \in \mathcal{K}_1 \setminus \mathcal{K}_2\}$ rekursiv aufzählbar.

Wir können jetzt eine Turingmaschine M'' konstruieren. Als Eingabe erhält sie $\langle M\rangle$, die Kodierung einer beliebigen Turingmaschine M. Danach werden die folgende Schritte von M'' ausgeführt:

1. M'' konstruiert aus M die Sprache $L_M \in \mathcal{L}(\mathcal{K}_1)$ mit Beschreibung $K \in \mathcal{K}_1$.
2. M'' konstruiert aus K die Turingmaschine M'.
3. M'' überprüft mit Hilfe von M', ob $L(K) = L_M \in \mathcal{L}(\mathcal{K}_2)$.
4. M'' hält und akzeptiert, wenn M' hält und akzeptiert, also wenn $L(K) = L_M \notin \mathcal{L}(\mathcal{K}_2)$.

Da laut den Voraussetzungen $L_M \in \mathcal{L}(\mathcal{K}_2)$ genau dann gilt, wenn $T(M)$ endlich ist, ist

$$T(M'') = \{\langle M\rangle \mid T(M) \text{ unendlich}\}.$$

Das ist aber ein Widerspruch zu Satz 2.6. In unserem Fall ist die Eigenschaft $\mathcal{P}$ die Eigenschaft Unendlichkeit und $L_{\mathcal{P}} = \{\langle M\rangle \mid T(M) \text{ unendlich}\}$. Für jedes $L \in \mathcal{P}$ ist aber jede endliche Teilmenge $L' \subset L$ nicht in $\mathcal{P}$ enthalten. Daher trifft Punkt 2. von Satz 2.6 nicht zu und $T(M'')$ ist keine rekursiv aufzählbare Sprache.

Insgesamt erhalten wir durch den Widerspruch, daß der Tradeoff zwischen $\mathcal{K}_1$ und $\mathcal{K}_2$ nichtrekursiv ist. □

Mit dem Ergebnis aus Lemma 2.8, daß die „invalid computations“ linear sind und einem zweiten Ergebnis über *INVALC*(M) werden wir die Methode zum Nachweis nichtrekursiver Tradeoffs anhand eines Beispiels verdeutlichen.

Lemma 2.10 *[Har80] Für jede Turingmaschine M ist die Sprache INVALC(M) in $\mathcal{L}(REG)$ genau dann, wenn $T(M)$ endlich ist.*

Beispiel 2.3 In diesem Beispiel wollen wir zeigen, daß der Tradeoff zwischen *LIN* und *REG* nichtrekursiv ist. Wir wissen schon, daß für jede Turingmaschine M eine Sprache, nämlich *INVALC*(M), effektiv konstruiert werden kann mit *INVALC*$(M) \in \mathcal{L}(LIN)$. Außerdem haben wir im letzten Lemma gesehen, daß *INVALC*$(M) \in \mathcal{L}(REG)$ genau dann gilt, wenn $T(M)$ endlich ist.

Wir können also Satz 2.7 anwenden und haben mit *INVALC*(M) die Sprache L_M, mit *LIN* das Beschreibungssystem $\mathcal{K}$ und mit *REG* das Beschreibungssystem $\mathcal{K}_2$. Insgesamt erhalten wir, daß gilt

$$REG \overset{nonrec}{\longleftarrow} LIN.$$

Kapitel 3

Verwandte Modelle

In der Vergangenheit wurden zahlreiche Modelle zur Erweiterung der generativen Mächtigkeit von kontextfreien Grammatiken vorgeschlagen. Bei jedem dieser Modelle wird die Einfachheit der kontextfreien Produktionen beibehalten, während die generative Mächtigkeit der kontextfreien Grammatik durch Kontrollstrukturen vergrößert wird. ET0L-Systeme, Matrixgrammatiken, programmierte Grammatiken und auch kontextfreie CD Grammatiksysteme sind solche Modelle. Bevor wir im nächsten Kapitel genau auf CD Grammatiksysteme eingehen, wenden wir uns den verwandten Modellen zu, die im folgenden noch häufiger benutzt werden.

3.1 ET0L-Systeme

L-Systeme wurden von Aristid Lindenmayer ursprünglich definiert, um biologische Prozesse zu beschreiben. Eine gute Einführung in L-Systeme und ihre mathematischen Grundlagen findet sich in [RS80]. Wir konzentrieren uns auf ET0L-Systeme, eine Form mit Terminalen und mehreren Tabellen, die mit den CD Grammatiksystemen eng verwandt ist.

Definition 3.1 *Ein **ET0L-System** ist ein $(n+3)$-Tupel*

$$G = (V, T, P_1, \ldots, P_n, \omega)$$

mit $n \geq 1$, wobei

1. *V das Alphabet ist,*
2. *$T \subseteq V$ das Terminalalphabet ist,*
3. *P_i mit $1 \leq i \leq n$ die Tabellen mit jeweils endlich vielen Produktionen aus $V \times V^*$ sind und*
4. *$\omega \in V^*$ der Startstring ist.*

Mit N bezeichnen wir die Menge $V \setminus T$ der Nichtterminale.

Für $1 \leq i \leq n$ und alle Symbole $a \in V$ existiert mindestens eine Regel der Form $a \to \alpha$ in P_i. Man nennt solche Tabellen auch komplett. Regeln der Form $a \to a$, $a \in V$ werden wir in Zukunft nicht mehr explizit angeben.

Definition 3.2 *Sei $G = (V, T, P_1, \ldots, P_n, \omega)$ ein ET0L-System. Für zwei Strings $\alpha, \alpha' \in V^*$ existiert ein Ableitungsschritt $\alpha \Rightarrow_{P_i} \alpha'$, wenn es Produktionen*

$$r_1, r_2, \ldots, r_p \in P_i$$

gibt mit $r_i = a_i \to \beta_i$ und $1 \leq i \leq p$, so daß

$$\alpha = a_1 a_2 \ldots a_p \text{ und } \alpha' = \beta_1 \beta_2 \ldots \beta_p.$$

Wir schreiben $\alpha \overset{*}{\Rightarrow} \alpha'$, wenn α' von α aus in endlich vielen Schritten (0 Schritte sind auch erlaubt) erreicht werden kann.

*Die vom ET0L-System G erzeugte **Sprache** ist*

$$L(G) = \{w \in T^* \mid \omega \overset{*}{\Rightarrow} w\}.$$

Bei einer kontextfreien Grammatik unterscheiden wir zwischen Nichtterminalen und Terminalen. Die Terminale einer kontextfreien Grammatik dürfen nicht durch eine Produktion überschrieben werden. Definiert man ein ET0L-System wie oben, ist es jedoch möglich, Terminale nicht identisch zu ersetzen. Durch eine einfache Konstruktion kann man aber aus jedem ET0L-System, das der obigen Definition folgt, ein ET0L-System erhalten, das Terminale und Nichtterminale im Sinne einer kontextfreien Grammatik hat.

Sei $G = (V, T, P_1, \ldots, P_n, w)$ ein ET0L-System. Wir definieren zuerst die Menge $N = \{A^{(1)} \mid \exists A \to \alpha \in P_i, 1 \leq i \leq n, A \in T, \alpha \neq A\}$ Als nächstes ersetzen wir in allen Tabellen die Terminale A durch $A^{(1)}$ und fügen Regeln der Form $A^{(1)} \to A$ zu den Tabellen hinzu. Es ist also $P_i' = \{A \to \alpha^{(1)} \mid A \in (V \setminus T), A \to \alpha \in P_i\} \cup \{A^{(1)} \to \alpha^{(1)} \mid A \in N, A \to \alpha \in P_i\} \cup \{A^{(1)} \to A \mid A \in N\}, 1 \leq i \leq n$. Dabei bezeichnet $\alpha^{(1)}$ den Homomorphismus, der bei Strings $\alpha \in V$ alle Zeichen aus $V \setminus N$ unverändert läßt und alle Zeichen $B \in N$ durch $B^{(1)}$ ersetzt. Weiterhin müssen wir die neu eingeführten Zeichen noch zur Menge der Variablen hinzufügen, es ist also $V' = V \cup N$. Wir haben jetzt ein ET0L-System $G' = (V', T, P_1', \ldots, P_n', w^{(1)})$ konstruiert, das dieselbe Sprache wie G erzeugt. Die Menge der Variablen teilt sich aber in Terminale T und Nichtterminale im Sinne einer kontextfreien Grammatik auf. Kein Terminal wird durch eine Regel nichtidentisch ersetzt. In Zukunft können wir also annehmen, daß ein ET0L-System Terminale und Nichtterminale im Sinne einer kontextfreien Grammatik enthält.

Als nächstes führen wir eine Einschränkung für ET0L-Systeme ein, die wir auch schon für kontextfreie Grammatiken definiert haben. Durch den endlichen Index erhalten wir eine echte Unterklasse der uneingeschränkten ET0L-Systeme.

Definition 3.3 *Sei $G = (V, T, P_1, \ldots, P_n, \omega)$ ein ET0L-System und $\alpha_0, \ldots, \alpha_r \in V^*$. Für eine Ableitung*

$$D : \alpha_0 \Rightarrow_{P_{i_1}} \alpha_1 \Rightarrow_{P_{i_2}} \ldots \Rightarrow_{P_{i_r}} \alpha_r$$

in G ist

$$Ind(D) = \max\{|\alpha_j|_N \mid 1 \leq j \leq r\}.$$

Eine Satzform α von G hat den Index

$$Ind(\alpha) = \min\{Ind(D) \mid D : \omega \stackrel{*}{\Rightarrow} \alpha\}.$$

*Der **Index von G** ist*

$$Ind(G) = \sup\{Ind(w) \mid w \in L(G)\}.$$

Ein ET0L-System hat einen endlichen Index, wenn es vom Index m für ein $m \geq 0$ ist.

Eine noch stärkere Einschränkung sind metalineare Produktionen. Auch hier folgen wir den Definitionen für kontextfreie Grammatiken.

Definition 3.4 *Ein ET0L-System $G = (V, T, P_1, \ldots, P_n, \omega)$ wird genau dann **m-linear** genannt, wenn es die folgenden Eigenschaften hat: $\omega = S \in V \setminus T$ kommt nicht auf der rechten Seite einer Produktion vor. Alle Produktionen sind in der Form*

$$S \to A_1 \ldots A_{m_0}, \qquad A \to uBv \qquad \text{oder} \qquad A \to w$$

*mit $A, B, A_1, \ldots, A_{m_0} \in N \setminus \{S\}$, $m_0 \leq m$ und $u, v, w \in T^*$. Ein ET0L-System wird **metalinear** genannt, wenn es m-linear für ein $m \geq 1$ ist.*

Wir nennen die Klasse der ET0L-Systeme *ET0L*, die Klasse der ET0L-Systeme vom Index m *ET0L-mFIN* und die Klasse der ET0L-Systeme von endlichem Index *ET0L-FIN*. Für die Klasse der m-linearen ET0L-Systeme schreiben wir *ET0L-mLIN* und für metalineare ET0L-Systeme *ET0L-METALIN*. Die entsprechenden Sprachklassen heißen dann $\mathcal{L}$(*ET0L*), $\mathcal{L}$(*ET0L-mFIN*), $\mathcal{L}$(*ET0L-FIN*), $\mathcal{L}$(*ET0L-mLIN*) und $\mathcal{L}$(*ET0L-METALIN*).

3.2 Matrixgrammatiken

Matrixgrammatiken wurden von Samuel Abraham 1965 definiert. Im Gegensatz zu kontextfreien Grammatiken bestehen Matrixgrammatiken aus Sequenzen von kontextfreien Regeln, den Matrizen. In einem Ableitungsschritt müssen alle Regeln einer solchen Matrix in der gegebenen Reihenfolge benutzt werden. Auf diese Weise können Matrixgrammatiken auch einige kontextsensitive Sprachen erzeugen. Ein umfassender Überblick über das Thema findet sich in [DP89].

Definition 3.5 *Eine kontextfreie **Matrixgrammatik** ist ein Viertupel*

$$G = (N, T, M, S),$$

wobei

1. N *eine endliche Menge von Nichtterminalen,*
2. T *eine endliche Menge von Terminalen,*
3. $S \in N$ *das Startsymbol und*
4. M *eine endliche Menge von Sequenzen der Form*

   $$m : (r_1, r_2, \ldots, r_{n_m})$$

   mit $n_m \geq 1$ *ist.*

Die Regeln $r_i : A \to \alpha$ *aus* $N \times (N \cup T)^*$ *sind kontextfrei.*

Definition 3.6 *Sei* $G = (N, T, M, S)$ *eine Matrixgrammatik. Für zwei Strings* $\alpha, \alpha' \in (N \cup T)^*$ *schreiben wir genau dann* $\alpha \Rightarrow \alpha'$*, wenn es Strings*

$$\alpha_0, \alpha_1, \ldots, \alpha_n \in (N \cup T)^*$$

und eine Sequenz

$$m : (r_1, r_2, \ldots, r_n) \in M \text{ mit } r_i = A_i \to \beta_i, 1 \leq i \leq n$$

gibt, so daß

$$\alpha_0 = \alpha,\ \alpha_n = \alpha' \text{ und } \alpha_{i-1} = \gamma_{i-1} A_i \gamma'_{i-1},\ \alpha_i = \gamma_{i-1} \beta_i \gamma'_{i-1}$$

für

$$\gamma_{i-1}, \gamma'_{i-1} \in (N \cup T)^* \text{ und } 1 \leq i \leq n.$$

$\alpha \Rightarrow \alpha'$ *wird ein Ableitungsschritt genannt und wir schreiben* $\alpha \overset{*}{\Rightarrow} \alpha'$*, wenn* α' *von* α *aus in endlich vielen Schritten (0 Schritte sind auch erlaubt) erreicht werden kann. Die von einer Matrixgrammatik* G *erzeugte* ***Sprache*** *ist*

$$L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}.$$

Auch hier führen wir die Einschränkung des endlichen Index ein.

Definition 3.7 *Sei* G *eine Matrixgrammatik und* $\alpha_0, \alpha_1, \ldots, \alpha_r \in (N \cup T)^*$*. Der Index einer Ableitung*

$$D : S = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_r$$

in G *ist*

$$Ind(D) = \max\{|\alpha_j|_N \mid 1 \leq j \leq r\}.$$

Für eine Satzform α *von* G *definieren wir den Index mit*

$$Ind(\alpha) = \min\{Ind(D) \mid D : S \Rightarrow^* \alpha\}.$$

Der ***Index von G*** *ist*

$$Ind(G) = \sup\{Ind(w) \mid w \in L(G)\}.$$

Eine Matrixgrammatik hat einen endlichen Index, wenn sie vom Index m *für ein* $m \geq 0$ *ist.*

Eine stärkere Einschränkung ist Metalinearität. Wir folgen auch hier der Definition für den kontextfreien Fall.

Definition 3.8 *Eine Matrixgrammatik $G = (N, T, M, S)$ wird genau dann **m-linear** genannt, wenn sie die folgenden Eigenschaften hat: S kommt nicht auf der rechten Seite einer Produktion vor. Alle Produktionen sind in der Form*

$$S \to A_1 \dots A_{m_0}, \qquad A \to uBv \qquad \text{oder} \qquad A \to w$$

*mit $A, B, A_1, \dots, A_{m_0} \in N \setminus \{S\}$, $m_0 \leq m$ und $u, v, w \in T^*$. Eine Matrixgrammatik wird **metalinear** genannt, wenn sie m-linear für ein $m \geq 1$ ist.*

Wir nennen die Klasse der Matrixgrammatiken *MAT* die Klasse der Matrixgrammatiken vom Index m *MAT-mFIN*, die Klasse der Matrixgrammatiken mit endlichem Index *MAT-FIN*. Außerdem schreiben wir für die Klasse der m-linearen Matrixgrammatiken *MAT-mLIN* und die Klasse der metalinearen Matrixgrammatiken *MAT-METALIN*. Die entsprechenden Sprachklassen bezeichnen wir mit $\mathcal{L}$(*MAT*), $\mathcal{L}$(*MAT-mFIN*), $\mathcal{L}$(*MAT-FIN*), $\mathcal{L}$(*MAT-mLIN*) und $\mathcal{L}$(*MAT-METALIN*).

3.3 Programmierte Grammatiken

Programmierte Grammatiken fallen genau wie Matrixgrammatiken in das Gebiet des „regulated rewriting“. Auch sie bestehen aus kontextfreien Regeln. Allerdings wird bei programmierten Grammatiken die als nächstes angewendete Produktion von der vorherigen bestimmt. Das Modell wurde in [Ros69] eingeführt. Darüber hinaus findet man einen umfassenden Überblick über programmierte Grammatiken in [DP89]. Eine besondere Form der programmierten Grammatiken sind „recurrent“ programmierte Grammatiken, die in [vS76] eingeführt wurden.

Definition 3.9 *Eine **programmierte Grammatik** ist ein Viertupel*

$$G = (N, T, P, S),$$

wobei

1. *N eine endliche Menge von Nichtterminalen,*
2. *T eine endliche Menge von Terminalen,*
3. *$S \in N$ das Startsymbol und*
4. *P eine endliche Menge von Tripeln der Form*

   $$r : A \to \alpha,\ \sigma(r),\ \psi(r)$$

   ist.

Dabei ist $Lab(P) = \{r \mid (r : A \to \alpha, \sigma(r), \psi(r)) \in P\}$. Die Regeln $r : A \to \alpha$ über $N \times (N \cup T)^*$ sind kontextfrei und $\sigma(r)$ und $\psi(r)$ sind zwei Teilmengen von $Lab(P)$.

Bei den programmierten Grammatiken ist jede Regel oder auch Produktion mit einer Bezeichnung und zwei Mengen, die aus solchen Bezeichnungen bestehen, verbunden. Für eine Produktion r wird $\sigma(r)$ auch das Erfolgsfeld und $\psi(r)$ das Fehlerfeld genannt. Im Erfolgsfeld von r sind gerade alle Bezeichnungen von Produktionen, die angewendet werden dürfen, nachdem r erfolgreich angewendet wurde. Im Fehlerfeld von r sind alle Bezeichnungen von Produktionen, die angewendet werden dürfen, nachdem r nicht angewendet werden konnte.

Definition 3.10 *Seien (α_1, r_1) und (α_2, r_2) in $(N \cup T) \times Lab(P)$. Wir schreiben $(\alpha_1, r_1) \Rightarrow (\alpha_2, r_2)$ genau dann, wenn*

1. *$\alpha_1 = \gamma A \gamma'$, $\alpha_2 = \gamma \beta \gamma'$, $\gamma, \gamma' \in (N \cup T)$, $(r_1 : A \to \beta, \sigma(r_1), \phi(r_1)) \in P$ und $r_2 \in \sigma(r_1)$ oder*
2. *$\alpha_1 = \alpha_2$, $(r_1 : A \to \beta, \sigma(r_1), \phi(r_1)) \in P$ kann nicht auf α_1 angewendet werden und $r_2 \in \phi(r_1)$.*

$\alpha_1 \Rightarrow \alpha_2$ wird ein Ableitungsschritt genannt und wir schreiben $\alpha \overset{*}{\Rightarrow} \alpha'$, wenn α' von α aus in endlich vielen Schritten (0 Schritte sind auch erlaubt) erreicht werden kann.

*Die von einer programmierten Grammatik G erzeugte **Sprache** ist*

$$L(G) = \{w \in T^* \mid (S, r_1) \overset{*}{\Rightarrow} (w, r_2),\ r_1, r_2 \in Lab(P)\}.$$

Ein Ableitungsschritt $(\alpha_1, r_1) \Rightarrow (\alpha_2, r_2)$ kann also genau dann stattfinden, wenn entweder r_1 auf α_1 angewendet werden kann und r_2 im Erfolgsfeld von r_1 ist, oder r_1 nicht auf α_1 angewendet werden kann und r_2 im Fehlerfeld von r_1 ist.

Die Klasse der programmierten Grammatiken bezeichnen wir mit (P, ac). Dabei bezeichnet ac ein nicht leeres Fehlerfeld und damit die Möglichkeit zu überprüfen, ob eine Produktion nicht angewendet werden kann. Diese Eigenschaft wird auch „appearence checking“ genannt. Ist $\phi(r) = \emptyset$ für alle $r \in P$, dann erhalten wir die Klasse der programmierten Grammatiken ohne „appearence checking“, für die wir P schreiben.

Die Klasse der „recurrent“ programmierten Grammatiken besteht aus allen programmierten Grammatiken, bei denen für alle Produktionen r gilt, daß r in seinem eigenen Erfolgsfeld enthalten ist und das Fehlerfeld entweder leer oder die gleiche Menge wie das Erfolgsfeld ist. Formal ist dann $r \in \sigma(r)$ und entweder $\phi(r) = \emptyset$ oder $\phi(r) = \sigma(r)$ für alle $r \in P$. Diese Klasse wird mit (RP, ac) bezeichnet. Die jeweils erzeugten Sprachklassen sind dann $\mathcal{L}(P, ac)$, $\mathcal{L}(P)$ und $\mathcal{L}(RP, ac)$.

Kapitel 4

CD Grammatiksysteme

Ein wichtiger Punkt in der klassischen Theorie der formalen Sprachen ist die Beschreibung von Sprachen durch Grammatiken oder Automaten. Einige dieser Modelle haben wir im letzten Kapitel kennengelernt. Im Gegensatz zu diesen Modellen, die aus einer einzelnen Komponente bestehen, werden wir jetzt ein System mit mehreren Komponenten betrachten. Ein Grammatiksystem ist eine Menge von Grammatiken, die auf eine genau definierte Art und Weise miteinander kooperieren.

Kontextfreie „cooperating distributed“ (kurz CD) Grammatiksysteme bestehen aus mehreren kontextfreien Grammatiken, die auch Komponenten genannt werden. CD Grammatiksysteme sind sequentielle Modelle. Das heißt, daß an einer gemeinsamen Satzform zu einem Zeitpunkt nur eine Komponente arbeitet. Diese Komponente wird auch als aktive Komponente bezeichnet. Auf welche Art und Weise die Komponenten zusammenarbeiten, bestimmt der Ableitungsmodus. Im t-Ableitungsmodus werden zum Beispiel so lange Produktionen aus einer Komponente benutzt, bis keine Produktion mehr anwendbar ist. Danach wird eine andere Komponente aktiv. Im $(= k)$-Modus werden genau k Produktionen aus einer Komponente angewendet. Danach wird die nächste Komponente aktiv.

Der erste Abschnitt dieses Kapitels enthält die notwendigen Definitionen. Der nächste Abschnitt geht auf die generative Mächtigkeit und andere Eigenschaften von CD Grammatiksystemen ein. Danach wird ein Überblick über die CD Grammatiksysteme betreffenden Ergebnisse aus der Beschreibungskomplexität gegeben.

Wir folgen bei den Definitionen vor allen Dingen [DPR97]. Die meisten Ergebnisse sind aus dem Standardwerk [CVDKP94], welches einen sehr guten Überblick über alle Aspekte von CD Grammatiksystemen bietet.

4.1 Definitionen und Beispiele

Nachdem wir alle verwandten Modelle im letzten Kapitel vorgestellt haben, können wir jetzt kontextfreie CD Grammatiksysteme definieren. Nach den nötigen Definitionen wird in einigen Beispielen ihre Arbeitsweise verdeutlicht.

Definition 4.1 *Ein kontextfreies* ***CD Grammatiksystem*** *ist ein* $(n+3)$*-Tupel*

$$\Gamma = (N, T, P_1, \ldots, P_n, S)$$

mit $n \geq 1$*, wobei*

1. N *eine endliche Menge von Nichtterminalen ist,*
2. T *eine endliche Menge von Terminalen ist,*
3. P_i *mit* $1 \leq i \leq n$ *endliche Mengen von Produktionen aus* $N \times (N \cup T)^*$ *sind und*
4. $S \in N$ *das Startsymbol ist.*

Wir schreiben

$$\beta_1 A \beta_2 \Rightarrow_{P_i} \beta_1 \alpha \beta_2,$$

wenn $A \to \alpha \in P_i$ ist und $\beta_1, \beta_2 \in (N \cup T)^*$ sind. Man sagt, daß die Produktion $A \to \alpha$ auf die ***Satzform*** $\beta_1 A \beta_2$ angewendet wird.

Außerdem benennen wir die Menge aller linken Seiten von Produktionen in P_i mit

$$dom(P_i) = \{A \in N \mid A \to \alpha \in P_i\}.$$

Die Produktionsmengen werden auch als ***Komponenten*** bezeichnet.

Wir werden jetzt sehen, wie die Komponenten benutzt werden, um Terminalwörter zu generieren. Die Art und Weise, wie die Komponenten zusammenarbeiten, bestimmt der Ableitungsmodus. Im folgenden betrachten wir verschiedene Ableitungsmodi. Sie werden in der nächsten Definition vorgestellt.

Definition 4.2 *Sei* $\Gamma = (N, T, P_1, \ldots, P_n, S)$ *ein CD Grammatiksystem, seien* $\alpha, \alpha' \in (N \cup T)^*$ *und seien* $k, k' \geq 1$*.*

1. *Für alle* i *mit* $1 \leq i \leq n$ *ist eine* ***k-Schritt Ableitung*** *in der* i*-ten Komponente definiert als*

   $$\alpha \overset{=k}{\Longrightarrow}_{P_i} \alpha'$$

   genau dann, wenn $\alpha_1, \ldots, \alpha_{k+1} \in (N \cup T)^*$ *so daß* $\alpha = \alpha_1$*,* $\alpha' = \alpha_{k+1}$ *und* $\alpha_j \Rightarrow_{P_i} \alpha_{j+1}$ *mit* $1 \leq j \leq k$*.*

2. *Für alle* i *mit* $1 \leq i \leq n$ *ist eine* ***höchstens k-Schritt Ableitung*** *in der* i*-ten Komponente definiert als*

   $$\alpha \overset{\leq k}{\Longrightarrow}_{P_i} \alpha'$$

   genau dann, wenn $\alpha \overset{=k'}{\Longrightarrow}_{P_i} \alpha'$ *für ein* $k' \leq k$*.*

3. *Für alle i mit $1 \leq i \leq n$ ist eine **mindestens k-Schritt Ableitung** in der i-ten Komponente definiert als*

   $$\alpha \overset{\geq k}{\Longrightarrow}_{P_i} \alpha'$$

   genau dann, wenn $\alpha \overset{=k'}{\Longrightarrow}_{P_i} \alpha'$ für ein $k' \geq k$.

4. *Für alle i mit $1 \leq i \leq n$ ist eine **$*$-Modus Ableitung** in der i-ten Komponente definiert als*

   $$\alpha \overset{*}{\Longrightarrow}_{P_i} \alpha'$$

   genau dann, wenn $\alpha \overset{=k'}{\Longrightarrow}_{P_i} \alpha'$ für ein beliebiges $k' \geq 0$.

5. *Für alle i mit $1 \leq i \leq n$ ist eine **terminierende Ableitung** in der i-ten Komponente definiert als*

   $$\alpha \overset{t}{\Longrightarrow}_{P_i} \alpha'$$

   genau dann, wenn $\alpha \overset{*}{\Longrightarrow}_{P_i} \alpha'$ ist und es kein $\beta \in (N \cup T)^*$ mit $\alpha' \Rightarrow_{P_i} \beta$ gibt.

Nachdem wir die verschiedenen Ableitungsmodi kennengelernt haben, können wir jetzt die von einem CD Grammatiksystem erzeugte Sprache definieren.

Definition 4.3 *Sei $f \in \{*, t, \leq k, = k, \geq k \mid k \geq 1\}$. Die von einem CD Grammatiksystem $\Gamma = (N, T, P_1, \ldots, P_n, S)$ im Ableitungsmodus f generierte **Sprache** ist*

$$\begin{aligned} L(\Gamma, f) \quad = \quad & \{w \in T^* \mid S \Rightarrow^f_{P_{i_1}} \alpha_1 \Rightarrow^f_{P_{i_2}} \ldots \Rightarrow^f_{P_{i_r}} \alpha_r = w, \\ & r \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq r\}. \end{aligned}$$

Wir nennen die Klasse aller (kontextfreien) CD Grammatiksysteme mit n Komponenten *CD$_n$-CF* und die Klasse aller CD Grammatiksysteme mit n Komponenten, die im Modus f arbeitet (*CD$_n$-CF*, f). Die entsprechende Sprachklasse heißt dann $\mathcal{L}$(*CD$_n$-CF*, f). Wenn die Anzahl der Komponenten nicht beschränkt ist, wird n weggelassen.

Bei einigen Konstruktionen werden wir statt einer Ableitung den daraus entstehenden Ableitungsbaum betrachten. Dieser Begriff wird jetzt definiert.

Definition 4.4 *Sei $\Gamma = (N, T, P_1, \ldots, P_n, S)$ ein CD Grammatiksystem. Ein markierter, geordneter Baum wird genau dann **Ableitungsbaum** von Γ genannt, wenn folgendes gilt:*

1. *Jeder Knoten hat eine Markierung, die ein Symbol aus $N \cup T \cup \{\varepsilon\}$ ist.*
2. *Die Markierung der Wurzel ist S.*

3. *Wenn ein innerer Knoten die Markierung A hat, dann muß $A \in N$ sein.*
4. *Wenn der Knoten n die Markierung A hat und die Knoten $n_1, n_2, \ldots, n_k$ geordnet von links nach rechts die Söhne von n mit den Markierungen $X_1, X_2, \ldots, X_k$ sind, dann muß eine Produktion $A \to X_1 X_2 \ldots X_k$ in einer Komponente von Γ vorkommen.*
5. *Wenn der Knoten n eine Markierung ε hat, dann ist n ein Blatt und der einzige Sohn seines Vaters.*

Die Markierungen an den Blättern eines Ableitungsbaums t von links nach rechts gelesen werden $yield(t)$ genannt. Ein Teilbaum eines Ableitungsbaums ist ein bestimmter Knoten mit all seinen Nachkommen.

Es soll hier noch angemerkt werden, daß jede Ableitung eines CD Grammatiksystems, die einem bestimmten Ableitungsmodus folgt, auch einem Ableitungsbaum entspricht. Allerdings gibt es auch Ableitungsbäume, zu denen es keine gültige Ableitung gibt.

Wir nennen die Ableitungsbäume, die einer Ableitung von Γ im Ableitungsmodus $f \in \{*, t, = k, \geq k, \leq k \mid k \geq 1\}$ entsprechen ***gültig*** im Bezug auf f. Wenn ein gültiger Ableitungsbaum t den $yield(t) = w \in T^*$ hat, nennen wir ihn ***gültigen vollständigen*** Ableitungsbaum.

Als nächstes wird anhand von drei Beispielen klar gemacht, wie CD Grammatiksysteme arbeiten. Zuerst werden wir ein CD Grammatiksystem im t-Modus betrachten, das eine nicht-kontextfreie Sprache erzeugt.

Beispiel 4.1 Die Sprache $L_1 = \{a^{2^n} \mid n \geq 1\}$ ist nicht kontextfrei. Betrachten wir nun das CD Grammatiksystem

$$\begin{aligned}
\Gamma_1 &= (\{S, A\}, \{a\}, P_1, P_2, P_3, S), \\
P_1 &= \{S \to AA\}, \\
P_2 &= \{A \to S\}, \\
P_3 &= \{A \to a\}.
\end{aligned}$$

Die Erzeugung eines Wortes beginnt immer mit der Komponente P_1. Dabei werden die zwei Nichtterminale AA eingeführt. Danach kann die Ableitung mit Komponente P_3 beendet werden. Alternativ können alle Nichtterminale A mit P_2 zu S umgewandelt werden und dann die Anzahl der Nichtterminale mit P_1 wieder verdoppelt werden. Eine Ableitung in Γ_1 im t-Modus sieht dann wie folgt aus:

$$\begin{aligned}
S \quad & \overset{t}{\Longrightarrow}_{P_1} AA \overset{t}{\Longrightarrow}_{P_2} SS \overset{t}{\Longrightarrow}_{P_1} AAAA \\
& (\overset{t}{\Longrightarrow}_{P_2} \ldots \overset{t}{\Longrightarrow}_{P_1} \ldots)^{i-1} A^{2^{i+1}} \overset{t}{\Longrightarrow}_{P_3} a^{2^{i+1}}.
\end{aligned}$$

Insgesamt wird die Sprache L_1 erzeugt.

Beispiel 4.2 Wir werden hier zeigen, wie das CD Grammatiksystem aus Beispiel 2.2 für $n \geq 1$ die Sprache $L_n = \bigcup_{r=1}^{n} L_{n,r}$ mit $L_{n,r} = \{a_r^{2i} \mid 1 \leq i \leq r\}$ erzeugt. Im Beispiel 2.2 wurde schon gezeigt, daß diese Sprache zwar kontextfrei ist, ein CD Grammatiksystem sie aber mit weniger Produktionen als eine kontextfreie Grammatik erzeugen kann. Ein CD Grammatiksystem für diese Sprache ist

$$\begin{aligned}
\Gamma_n &= (\{S, A\}, \{a_1, \ldots, a_n\}, P_1, \ldots, P_{n+1}, S), \\
P_1 &= \{A \to a_1\}, \\
&\ldots \\
P_n &= \{A \to a_n\}, \\
P_{n+1} &= \{S \to A^{2i} \mid 1 \leq i \leq n\}.
\end{aligned}$$

Γ_n arbeitet im t-Modus und wendet von S aus zuerst eine Produktion aus P_{n+1} an. Wir erhalten die Satzform A^{2i} für ein i, $1 \leq i \leq n$. Danach wird eine der Komponenten $P_1, \ldots, P_n$ aktiv. Wenn Komponente P_r aktiv wird, generiert Γ_n das Wort a_r^{2i}. Insgesamt ist $L(\Gamma_n, t) = L_n$.

Als nächstes werden wir ein Beispiel für den $(= k)$- und den $(\geq k)$-Modus sehen.

Beispiel 4.3 Hier wird die Sprache $L_2 = \{(a^n b^n)^k \mid n \geq 1\}$ erzeugt. Dazu verwenden wir das CD Grammatiksystem

$$\begin{aligned}
\Gamma_2 &= (\{S, A_1, A_2, \ldots, A_k\}, \{a\}, P_1, P_2, S), \\
P_1 &= \{S \to S, S \to A_1 A_2 \ldots A_k\} \cup \{A_i' \to A_i \mid 1 \leq i \leq k\}, \\
P_2 &= \{A_i \to a A_i' b \mid 1 \leq i \leq k\} \cup \{A_i \to ab \mid 1 \leq i \leq k\}.
\end{aligned}$$

Die Erzeugung eines Wortes im $(=k)$-Modus beginnt mit der Komponente P_1. Dabei wird die Satzform $A_1 A_2 \ldots A_k$ eingeführt. Mit der Produktion $S \to S$ wird sicher gestellt, daß genau k Produktionen angewendet werden können. Danach muß Komponente P_2 aktiv werden. Es können nur Produktionen der Form $A_i \to a A_i' b$ oder nur Produktionen der Form $A_i \to ab$ angewendet werden. Mischt man die beiden Arten von Produktionen, ist es danach nicht mehr möglich, genau k Produktionen anzuwenden, da weniger Nichtterminale in der Satzform vorhanden sind. Falls nur Produktionen der Form $A_i \to a A_i' b$ benutzt wurden, kann danach mit Komponente P_1 jedes markierte Nichtterminal (A_i') wieder durch ein unmarkiertes Nichtterminal (A_i) ersetzt werden. Eine Ableitung in Γ_2 im $(=k)$-Modus sieht dann wie folgt aus:

$$\begin{aligned}
S \quad & \overbrace{\Rightarrow_{P_1} S(\Rightarrow_{P_1} \ldots)^{k-2} \Rightarrow_{P_1}}^{=k} A_1 A_2 \ldots A_k \\
& \overset{=k}{\Longrightarrow}_{P_2} a A_1' b a A_2' b \ldots a A_k' b \overset{=k}{\Longrightarrow}_{P_1} a A_1 b a A_2 b \ldots a A_k b \\
& (\overset{=k}{\Longrightarrow}_{P_2} \ldots \overset{=k}{\Longrightarrow}_{P_1} \ldots)^{i-1} a^i A_1 b^i a^i A_2 b^i \ldots a^i A_k b^i \overset{=k}{\Longrightarrow}_{P_2} (a^{i+1} b^{i+1})^k.
\end{aligned}$$

Insgesamt wird die Sprache L_2 erzeugt.

Man kann das gleiche CD Grammatiksystem auch im $(\geq k)$-Modus benutzen. Nachdem mit der ersten Anwendung von P_1 die Satzform $A_1 A_2 \ldots A_k$ erzeugt wurde, können nie mehr als k Produktionen in einer Komponente angewendet werden. Die erzeugte Sprache bleibt also gleich.

4.2 Generative Mächtigkeit

Mit einigen der vorgestellten Modi erreichen auch CD Grammatiksysteme nur die Mächtigkeit von kontextfreien Grammatiken. Dies ist auch der Fall, wenn nur eine Komponente zugelassen ist. Die folgenden Resultate werden in [CVDKP94] gezeigt. Wir führen die Beweise hier kurz auf, um einige Argumente in späteren Beweisen benutzen zu können.

Satz 4.1 *Es gilt*

1. $\mathcal{L}(CF) = \mathcal{L}(CD_1\text{-}CF, d)$ *für* $d \in \{t, *, =k, \geq k, \leq k \mid k \geq 1\}$,
2. $\mathcal{L}(CF) = \mathcal{L}(CD\text{-}CF, f)$ *für* $f \in \{*, = 1, \geq 1, \leq k \mid k \geq 1\}$.

Beweis: 1. Wenn nur eine Komponente vorhanden ist, entspricht der t-Ableitungsmodus in einem CD Grammatiksystem der Ableitung eines Wortes in einer kontextfreien Grammatik. Daher gilt $\mathcal{L}(CF) = \mathcal{L}(CD_1\text{-}CF, t)$.

Sei jetzt $d \in \{= k, \geq k \mid k \geq 2\}$, L eine kontextfreie Sprache und $G = (N, T, P, S)$ eine kontextfreie Grammatik mit $L(G) = L$. Das Grammatiksystem $\Gamma = (N, T, P \cup \{A \to A \mid A \in N\}, S)$ erzeugt L in jedem Modus d, da durch die Produktionen $A \to A$ erreicht werden kann, daß die Anzahl der angewendeten Produktionen bei der Erzeugung eines Wortes ein Vielfaches von k ist.

Sei $\Gamma = (N, T, P_1, S)$ ein CD Grammatiksystem mit einer Komponente. Wir betrachten die kontextfreie Grammatik

$$G = (N, T \cup \{\$\}, \{A \to \alpha\$ \mid A \to \alpha \in P_1\}, S),$$

dabei ist $\$ \notin N \cup T$. Wenn $w \in L(G)$ und $|w|_\$ = r$, dann wurde w mit r Ableitungsschritten erzeugt. Wir definieren jetzt die reguläre Menge

$$K = \{w \mid w \in (T \cup \{\$\})^*,\ |w|_\$ = ki,\ i \geq 1\}.$$

und den Homomorphismus h mit $h(a) = a$ für $a \in T$ und $h(\$) = \varepsilon$. Man kann sehen, daß $h(L(G) \cap K) = L(\Gamma, = k)$. Wegen der Abschlußeigenschaften kontextfreier Sprachen ist also auch $L(\Gamma, = k)$ kontextfrei. Definiert man jetzt

$$K' = \{w \mid w \in (T \cup \{\$\})^*,\ |w|_\$ \geq i,\ i \geq k\},$$

dann erhält man $h(L(G) \cap K') = L(\Gamma, \geq k)$ und deshalb ist auch $L(\Gamma, \geq k)$ eine kontextfreie Sprache.

Für die restlichen Ableitungsmodi folgt 1. aus 2..

2. Da jede kontextfreie Grammatik auch als Grammatiksystem mit einer Komponente gesehen werden kann, gilt $\mathcal{L}(CF) \subseteq \mathcal{L}(CD\text{-}CF, f)$. Wir müssen also nur die Inklusion in die andere Richtung zeigen. Sei $\Gamma = (N, T, P_1, \ldots, P_n, S)$ ein CD Grammatiksystem. Wir konstruieren die kontextfreie Grammatik $G = (N, T, \bigcup_{1 \leq i \leq n} P_i, S)$, die $L(\Gamma, f)$ erzeugt. Da man bei allen Modi $f \in \{*, = 1, \geq 1, \leq k \mid k \geq 1\}$ auch genau eine Produktion in der aktiven Komponente anwenden kann, kann jedes Wort aus $L(G)$ in Γ im Modus f abgeleitet werden. Es ist also $L(G) \subseteq L(\Gamma, f)$. Man kann allerdings auch jedes Wort aus $L(\Gamma, f)$ in G ableiten, also gilt auch $L(\Gamma, f) \subseteq L(G)$. □

Die eben behandelten Klassen haben wegen ihrer Äquivalenz zu kontextfreien Sprachen auch die gleichen Abschlußeigenschaften.

Als nächstes wenden wir uns der generativen Mächtigkeit von CD Grammatiksystemen im t-Modus zu. Der folgende Satz mit Beweis ist aus [DPR97] und [CVD90] übernommen. Nur der zweite Teil des Beweises wird hier konstruktiv geführt, damit in den nächsten Abschnitten die Form der Produktionen genauer betrachtet werden kann.

Satz 4.2 $\mathcal{L}(ET0L) = \mathcal{L}(CD\text{-}CF, t) = \mathcal{L}(CD_3\text{-}CF, t)$.

Beweis: Zuerst werden wir zeigen, daß sich aus einem beliebigen CD Grammatiksystem im t-Modus ein CD Grammatiksystem mit nur drei Komponenten konstruieren läßt. Sei also $\Gamma = (N, T, P_1, \ldots, P_n, S)$ ein kontextfreies CD Grammatiksystem. Wir konstruieren nun $\Gamma' = (N', T, P_1', P_2', P_3', S^{(1)})$ und benutzen dabei die Homomorphismen (i) für $1 \leq i \leq n$ wie im Kapitel 2 definiert:

$$
\begin{aligned}
N' &= \{A^{(i)} \mid A \in N,\, 1 \leq i \leq n\},\\
P_1' &= \{A^{(i)} \to \alpha^{(i)} \mid A \to \alpha \in P_i,\, 1 \leq i \leq n\},\\
P_2' &= \{A^{(i)} \to A^{(i+1)} \mid A \in N,\, 1 \leq i < n,\, i \text{ ungerade}\} \cup Z_2\\
P_3' &= \{A^{(i)} \to A^{(i+1)} \mid A \in N,\, 1 \leq i < n,\, i \text{ gerade}\} \cup Z_3
\end{aligned}
$$

mit

$$
\begin{aligned}
Z_2 &= \{A^{(n)} \to A^{(1)} \mid A \in N\}, \text{ wenn } n \text{ ungerade, } \emptyset \text{ sonst,}\\
Z_3 &= \{A^{(n)} \to A^{(1)} \mid A \in N\}, \text{ wenn } n \text{ gerade, } \emptyset \text{ sonst.}
\end{aligned}
$$

Die Regeln aus P_i werden von P_1' und den Nichtterminalen der Form $A^{(i)}$ simuliert. Die Komponenten P_2' und P_3' verändern nur die Kennzeichnung der Nichtterminale und zwar für alle Nichtterminale in einer Satzform. Regeln aus P_i und P_j für $i \neq j$ können also nicht vermischt werden. Deshalb ist $L(\Gamma, t) = L(\Gamma', t)$.

Als nächstes zeigen wir, daß man aus einem CD Grammatiksystem mit drei Komponenten ein ET0L-System konstruieren kann. Sei also $\Gamma = (N, T, P_1, P_2, P_3, S)$ ein CD Grammatiksystem mit drei Komponenten. Das ET0L-System $G = (V, T, Q_1, Q_2, Q_3, Q_4, Q_5, S^{(0)})$ erzeugt $L(\Gamma, t)$. Dabei verwenden wir wieder die Homomorphismen (i) wie im Kapitel 2 definiert für $0 \leq i \leq 3$. Weiterhin ist $F \notin (N \cup T)$ ein neues Symbol und

$$
\begin{aligned}
V &= \{A^{(i)} \mid A \in N,\, 0 \leq i \leq 3\} \cup \{F\} \cup T, \\
Q_1 &= \{A^{(i)} \to \alpha^{(i)} \mid A \to \alpha \in P_i,\, 1 \leq i \leq 3\} \cup \\
&\quad \{B^{(i)} \to B^{(i)} \mid B \notin dom(P_i),\, 1 \leq i \leq 3\} \cup \\
&\quad \{a \to a \mid a \in T\} \cup \{F \to F\}, \\
Q_2 &= \{A^{(i)} \to A^{(0)} \mid A \notin dom(P_i),\, 1 \leq i \leq 3\} \cup \\
&\quad \{A^{(i)} \to F \mid A \in dom(P_i),\, 1 \leq i \leq 3\} \cup \\
&\quad \{a \to a \mid a \in T\} \cup \{F \to F\}, \\
Q_3 &= \{A^{(0)} \to A^{(1)} \mid A \in N\} \cup \\
&\quad \{a \to a \mid a \in T\} \cup \{F \to F\}, \\
Q_4 &= \{A^{(0)} \to A^{(2)} \mid A \in N\} \cup \\
&\quad \{a \to a \mid a \in T\} \cup \{F \to F\}, \\
Q_5 &= \{A^{(0)} \to A^{(3)} \mid A \in N\} \cup \\
&\quad \{a \to a \mid a \in T\} \cup \{F \to F\}.
\end{aligned}
$$

Das ET0L-System G beginnt wie Γ mit dem Nichtterminal S, allerdings mit der markierten Form $S^{(0)}$. Jetzt wird die Komponente von Γ ausgesucht, die G simulieren soll. Mit Q_3 werden alle Nichtterminale mit einer (1) markiert, durch Q_4 mit einer (2) und durch Q_5 mit einer (3). Danach können nur die Tabellen Q_1 und Q_2 benutzt werden. Q_1 simuliert eine Ableitung in der Komponente P_1, wenn die Nichtterminale mit (1) markiert sind, eine Ableitung in P_2 bei Markierung (2) oder eine Ableitung in P_3 bei Markierung (3). Durch die Markierung der Nichtterminale können die Produktionen verschiedener Komponenten nicht vermischt werden. Mit Q_2 werden wieder alle Nichtterminale mit (0) markiert. Das ist allerdings nur möglich, wenn die Nichtterminale vorher mit (i) markiert waren und kein Nichtterminal aus $dom(P_i)$ in der Satzform vorhanden ist. Ansonsten wird das Nichtterminal F eingeführt, das die Ableitung blockiert. Insgesamt wird so der t-Ableitungsmodus, in dem Γ arbeitet, simuliert. Es ist also $L(\Gamma, t) = L(G)$.

Als letztes muß noch gezeigt werden, daß man aus einem ET0L-System ein CD Grammatiksystem konstruieren kann. Sei $G = (V, T, P_1, \ldots, P_n, \omega)$ ein ET0L-System. Wir können, wie im Abschnitt über ET0L-Systeme gezeigt, annehmen, daß die Menge der aktiven Symbole und die Menge der Nichtterminale übereinstimmt. Seien weiterhin $S, F \notin V$ neue Nichtterminale. Wir definieren jetzt das CD Grammatiksystem $\Gamma = (N, T, P_1, P_2, P_3, S)$ und benutzen wieder Homomorphismen (i) für $0 \leq i \leq n$, die wie im Kapitel 2 definiert sind. Als die

Menge der Nichtterminale wird dabei $V \setminus T$ angenommen. Außerdem sind

$$\begin{aligned} N &= \{A^{(i)} \mid A \in V \setminus T, 0 \leq i \leq n\} \cup \{S, F\}, \\ P_1 &= \{A^{(i)} \to \alpha^{(0)} \mid A \to \alpha \in P_i, 1 \leq i \leq n\} \cup \{S \to \omega^{(0)}\}, \\ P_2 &= \{A^{(i)} \to A^{(i+1)} \mid A \in N, 0 \leq i < n, i \text{ gerade}\} \\ , P_3 &= \{A^{(i)} \to A^{(i+1)} \mid A \in N, 0 \leq i < n, i \text{ ungerade}\}. \end{aligned}$$

Zuerst muß P_1 angewendet werden und wir erhalten die Satzform $\omega^{(0)}$. Mit den Komponenten P_2 und P_3 wird die jeweils zu simulierende Tabelle ausgewählt. Danach ist wieder Komponente P_1 aktiv. Sie kann nur Regeln aus der ausgewählten Tabelle anwenden und zwar für jedes Symbol genau eine. Insgesamt wird also die Satzform so verändert, wie ein Ableitungsschritt von G sie verändern würde. Danach muß wieder mit P_2 und P_3 eine neue Tabelle ausgewählt werden. Eine Ableitung endet, wenn nur noch Nichtterminale vorhanden sind. Γ erzeugt also genau die Wörter, die auch G generieren kann. □

In [DPR97] wird außerdem gezeigt, daß alle CD Grammatiksysteme im t-Modus mit nur zwei Komponenten kontextfreie Sprachen erzeugen. Das eben erhaltene Ergebnis ist also optimal.

Da die Klasse $\mathcal{L}(CD\text{-}CF, t)$ äquivalent zu $\mathcal{L}(ET0L)$ ist, ist sie auch unter den gleichen Operationen abgeschlossen. Nach [RS80] sind das Vereinigung, Konkatenation, Schnitt mit regulären Mengen, Kleenesche Hülle, Homomorphismus und inverser Homomorphismus. Die Sprachklasse $\mathcal{L}(CD\text{-}CF, t)$ ist also eine volle AFL.

Als nächstes wird die generative Mächtigkeit der Ableitungsmodi $(= k)$ und $(\geq k)$ für $k \geq 2$ beschrieben. Die Beweise finden sich in [CVDKP94] und werden hier kurz aufgeführt, um die Argumente in folgenden Kapiteln weiter benutzen zu können.

Satz 4.3 *Es gilt*

$$\mathcal{L}(CD_1\text{-}CF, f) \subset \mathcal{L}(CD_2\text{-}CF, f) \subseteq \mathcal{L}(CD\text{-}CF, f) \subseteq \mathcal{L}(MAT)$$

für $f \in \{=k, \geq k \mid k \geq 2\}$.

Beweis: In Beispiel 4.3 wird eine nicht-kontextfreie Sprache mit einem CD Grammatiksystem mit zwei Komponenten im $(= k)$-Modus erzeugt. Daraus folgt die erste Inklusion. Die zweite Inklusion folgt aus der Definition.

Sei $\Gamma = (N, T, P_1, \ldots, P_n, S)$ ein CD Grammatiksystem. Wir konstruieren jetzt die Matrixgrammatik $G = (N, T, M, S)$. Dabei besteht M aus allen Matrizen

$$m = (p_1, \ldots, p_k),$$

so daß es ein i, $1 \leq i \leq n$, gibt mit $p_j \in P_i$ für $1 \leq j \leq k$. Jede Anwendung einer Matrix aus G entspricht also einer $(= k)$-Ableitung in Γ. Es gilt also $\mathcal{L}(CD_n\text{-}CF, = k) \subseteq \mathcal{L}(MAT)$. Wenn wir jetzt eine Matrixgrammatik $G' = (N, T, M', S)$ konstruieren, bei der M' aus allen Matrizen

$$m = (p_1, \ldots, p_u)$$

besteht, so daß es ein $i, 1 \leq i \leq n$ gibt mit $p_j \in P_i$ für $1 \leq j \leq u$ und $k \leq u \leq 2k-1$, dann kann jede Anwendung einer Matrix in G' durch eine $(\geq k)$-Schritt Ableitung simuliert werden. Außerdem besteht eine $(\geq k$-Ableitung aus $lk+v$ Schritten, wobei $l \geq 1$ und $1 \leq v < k$. Jede $(\geq k)$ Ableitung in Γ kann also durch eine oder mehrere Matrizen in G' simuliert werden. Insgesamt ist $L(\Gamma, \geq k) = L(G')$ und damit gilt $\mathcal{L}(CD_n\text{-}CF, \geq k) \subseteq \mathcal{L}(MAT)$. □

Satz 4.4 *[CVDKP94] Es gilt für $k \geq 2$*

1. $\mathcal{L}(CD\text{-}CF, = k) \subseteq \mathcal{L}(CD\text{-}CF, = k+1)$,
2. $\mathcal{L}(CD\text{-}CF, \geq k) \subseteq \mathcal{L}(CD\text{-}CF, \geq k+1)$.

In [GP90] und [DPV95] werden unter anderem auch die Abschlußeigenschaften der $(=k)$- und $(\geq k)$-Ableitungsmodi besprochen. Die Klassen $\mathcal{L}(CD\text{-}CF, = k)$ und $\mathcal{L}(CD\text{-}CF, \geq k)$ sind für $k \geq 2$ danach abgeschlossen unter Vereinigung, Konkatenation, Homomorphismus, inversem Homomorphismus und Schnitt mit regulären Mengen. Sie sind jedoch nicht unter Schnitt und Komplement abgeschlossen.

4.3 Beschreibungskomplexität

Wenn man die Beschreibungskomplexität von CD Grammatiksystemen betrachtet, stellt sich die Frage, ob es möglich ist, für bestimmte Sprachklassen „minimale“ Beschreibungen zu finden. Außerdem haben wir schon gesehen, daß CD Grammatiksysteme eine größere generative Mächtigkeit als kontextfreie Grammatiken haben. Es bleibt also herauszufinden, ob es Fälle gibt, in denen CD Grammatiksysteme kontextfreie Sprachen besser beschreiben können als kontextfreie Grammatiken.

In diesem Kapitel werden wir sehen, daß in einigen Fällen nicht nur die Anzahl der Komponenten beschränkt werden kann, sondern auch die Anzahl der Produktionen in einer Komponente, also das Maß $cProd$. Außerdem kann man verschiedene Aussagen darüber treffen, ob CD Grammatiksysteme kontextfreie Sprachen besser beschreiben können als kontextfreie Grammatiken.

Dieser Abschnitt soll die bekannten Ergebnisse nur kurz zusammenfassen. Alle Details finden sich in [DP91], [BR02] und [PDS94].

Als erstes gehen wir auf die optimale Größe eines CD Grammatiksystems ein. Dabei werden die Anzahl der Komponenten und die maximale Anzahl von Produktionen in einer Komponente betrachtet. Um die so erhaltenen Sprachklassen besser darstellen zu können, brauchen wir die folgende Definition.

Definition 4.5 *Im folgenden wird die Klasse der kontextfreien CD Grammatiksysteme mit n Komponenten und höchstens m Produktionen in einer Komponente $CD_{n,m}$-CF genannt. Die Sprachklasse, die diese Klasse von Grammatiksystemen im Ableitungsmodus $f \in \{*, t, \leq k, = k, \geq k \mid k \geq 1\}$ erzeugt, heißt*

dann $\mathcal{L}(CD_{n,m}\text{-}CF, f)$. Ist n oder m nicht nach oben beschränkt schreiben wir an der entsprechenden Stelle ∞.

Jetzt können wir die Ergebnisse aus [PDS94] zusammenfassen.

Für die Modi $f \in \{*, =1, \geq 1, \leq k \mid k \geq 1\}$ gilt

$$\mathcal{L}(CD_{\infty,\infty}\text{-}CF, f) = \mathcal{L}(CD_{\infty,1}\text{-}CF, t).$$

Wenn man sich mit den übrigen k-Ableitungsmodi beschäftigt, erhält man die folgende Schranke für $cProd$.

$$\begin{aligned}
\mathcal{L}(CD_{\infty,\infty}\text{-}CF, =k) &= \mathcal{L}(CD_{\infty,k}\text{-}CF, =k), \quad k \geq 2, \\
\mathcal{L}(CD_{\infty,\infty}\text{-}CF, \geq k) &= \mathcal{L}(CD_{\infty,(2k-1)}\text{-}CF, \geq k), \quad k \geq 2.
\end{aligned}$$

Schon im Abschnitt über die generative Mächtigkeit von kontextfreien CD Grammatiksystemen wurde gezeigt, daß

$$\mathcal{L}(CD_{\infty,\infty}\text{-}CF, t) = \mathcal{L}(CD_{3,\infty}\text{-}CF, t)$$

gilt. Auch $cProd$ kann im Fall des t-Modus auf eine Konstante beschränkt werden und es gilt

$$\mathcal{L}(CD_{\infty,\infty}\text{-}CF, t) = \mathcal{L}(CD_{\infty,5}\text{-}CF, t).$$

Wenn man jetzt einen der beiden Parameter $cProd$ oder Deg festhält, erhält man sowohl im t-Modus als auch im $*$-Modus eine unendliche Hierarchie über den anderen Parameter. Es gilt also für $f \in \{*, t\}$ und $n, m \geq 1$:

$$\begin{aligned}
\mathcal{L}(CD_{n,m}\text{-}CF, f) &\subset \mathcal{L}(CD_{n,m+1}\text{-}CF, f), \\
\mathcal{L}(CD_{n,m}\text{-}CF, f) &\subset \mathcal{L}(CD_{n+1,m}\text{-}CF, f).
\end{aligned}$$

Als nächstes wenden wir uns der Frage zu, ob kontextfreie CD Grammatiksysteme kontextfreie Sprachen besser darstellen können als kontextfreie Grammatiken. In [DP91] wird diese Frage in Bezug auf die Maße Var, $mProd$ und $Symb$ behandelt. In [BR02] werden einige der Ergebnisse verbessert und auch das Maß $cVar$ betrachtet.

Insgesamt sind die Ergebnisse für die Modi $\{t, =k, \geq k \mid k \geq 2\}$ für alle Maße sehr ähnlich. Es gibt unendlich viele Fälle, in denen man kontextfreie Sprachen mit CD Grammatiksystemen besser beschreiben als mit kontextfreien Grammatiken. Für die Maße Var, $mProd$ und $cVar$ gibt es sogar keine Funktion, die die Vergrößerung der Beschreibung beschränkt, wenn man von einem CD Grammatiksystem zu einer kontextfreien Grammatik wechselt.

Die Ergebnisse der Vergleiche mit kontextfreien Grammatiken im Bezug auf die verschiedenen Maße finden sich in Tabelle 4.1. Die Ergebnisse gelten für $a \in \{*, =1, \geq 1, \leq l \mid l \geq 1\}$ und $k \geq 2$.

	a	t	$=k$	$\geq k$
Var	$=$	$<_4$	$<_4$	$<_4$
$cVar$	$=$	$<_4$	$<_4$	$<_4$
$Prod$	$=$	$<_4$	$<_4$	$<_4$
$Symb$	$=$	$<_3$	$<_3$	$<_3$

Tabelle 4.1: Vergleich kontextfreier Grammatiken mit CD Grammatiksystemen für $a \in \{*, = 1, \geq 1, \leq l \mid l \geq 1\}$ und $k \geq 2$

4.4 Zusammenfassung

In diesem Kapitel haben wir kontextfreie CD Grammatiksysteme eingeführt und die Funktionsweise ausführlich mit Beispielen verdeutlicht. Danach wurde die generative Mächtigkeit betrachtet. Dabei fällt als erstes auf, daß nur in einigen Ableitungsmodi, nämlich in den Modi $\{t, = k, \geq k \mid k \geq 2\}$, die generative Mächtigkeit von CD Grammatiksystemen größer ist als die von kontextfreien Grammatiken. In Zukunft werden diese Modi daher auch die starken Ableitungsmodi genannt und die anderen als schwache Ableitungsmodi bezeichnet. Im Gegensatz zum t-Modus, dessen generative Mächtigkeit genau bestimmt werden kann, auch im Hinblick auf die Anzahl der Komponenten, bleiben bei den $(= k)$- und $(\geq k)$-Modi für $k \geq 2$ einige Fragen offen. Man weiß zum Beispiel nicht, ob die Hierarchie über die Anzahl der Komponenten echt ist oder an einer Stelle zusammenfällt. Außerdem ist ungeklärt, ob die Hierarchie über k echt ist oder an einer Stelle zusammenfällt.

Im Abschnitt über die Beschreibungskomplexität von kontextfreien CD Grammatiksystemen konnte man sehen, daß die starken Modi bei der Beschreibung von kontextfreien Sprachen unendlich oft besser sind als kontextfreie Grammatiken. Das gilt für viele der definierten Maße.

Außerdem konnte im Fall des t-Modus die maximale Anzahl von Produktionen $cProd$ und auch die maximale Anzahl von Komponenten Deg beschränkt werden. Wenn man einen der beiden Parameter $cProd$ oder Deg festhält, erhält man im t- und im $*$-Modus eine unendliche Hierarchie.

Kapitel 5

Hybride CD Grammatiksysteme

Hybride CD Grammatiksysteme können als Generalisierung von CD Grammatiksystemen verstanden werden. Man unterscheidet zwischen externer Hybridisierung, bei der jeder Komponente ein eigener Ableitungsmodus zugeordnet wird und interner Hybridisierung, bei der verschiedene Ableitungsmodi zu einem neuen Modus kombiniert werden. Die Definition der extern hybriden CD Grammatiksysteme ist etwas natürlicher als das ursprüngliche Modell. Einen guten Einblick in das Thema geben zum Beispiel [Mit93], [BH00] und [FHF01].

Da wir uns in dieser Arbeit nur mit extern hybriden CD Grammatiksystemen beschäftigen, werden wir sie im folgenden hybride CD Grammatiksysteme nennen. Sie unterscheiden sich von den vorher definierten CD Grammatiksystemen dadurch, daß jeder Komponente ein eigener Ableitungsmodus zugeordnet wird. Außerdem werden wir einen neuen Ableitungsmodus betrachten, der nur bei hybriden CD Grammatiksystemen benutzt werden kann. Während die Ergebnisse der generativen Mächtigkeit in [BH00], [FHF01] und [Mit93] zu finden sind, wurden die Ergebnisse über die Beschreibungskomplexität von hybriden CD Grammatiksystemen in [Sun03] veröffentlicht.

5.1 Definitionen und Beispiel

Voraussetzung für hybride CD Grammatiksysteme sind die Definitionen aus Kapitel 4. Die folgenden weiteren Definitionen werden auch benötigt.

Als erstes führen wir einen weiteren Ableitungsmodus ein. Eine Komponente P_i arbeitet im $\overline{t}$-Modus, wenn nach Beendigung ihrer Ableitungsschritte noch mindestens ein Nichtterminal vorhanden ist, das in $dom(P_i)$ ist. Auf diese Weise wird verhindert, daß eine Ableitung ausschließlich durch eine Komponente ausgeführt wird.

Definition 5.1 *Sei $\Gamma = (N, T, P_1, \ldots, P_n, S)$ ein CD Grammatiksystem und seien $\alpha, \alpha' \in (N \cup T)^*$. Für alle i mit $1 \leq i \leq n$ ist eine **nichtterminierende Ableitung** in der i-ten Komponente definiert als*

$$\alpha \overset{\overline{t}}{\Rightarrow}_{P_i} \alpha'$$

genau dann, wenn $\alpha \overset{*}{\Rightarrow}_{P_i} \alpha'$ gilt und es ein $\beta \in (N \cup T)^*$ mit $\alpha' \Rightarrow_{P_i} \beta$ gibt.

Da man mit dem $\overline{t}$-Ableitungsmodus keinen Terminalstring erreichen kann, wird er nur bei hybriden CD Grammatiksystemen benutzt.

Definition 5.2 *Sei $M_h = \{*, t, \overline{t}, \leq k, = k, \geq k \mid k \geq 1\}$. Ein **(extern) hybrides CD Grammatiksystem** ist ein $(n+3)$-Tupel*

$$\Gamma = (N, T, (P_1, f_1), (P_2, f_2), \ldots, (P_n, f_n), S),$$

wobei $n \geq 1$ ist, $(N, T, P_1, \ldots, P_n, S)$ ein normales CD Grammatiksystem ist und $f_1, \ldots, f_n \in M_h$ sind. In der i-ten Komponente von Γ wird der Ableitungsmodus f_i benutzt.

Wenn alle Komponenten im gleichen Ableitungsmodus arbeiten, erhalten wir ein normales CD Grammatiksystem.

Definition 5.3 *Die von einem (extern) hybriden CD Grammatiksystem*

$$\Gamma = (N, T, (P_1, f_1), (P_2, f_2), \ldots, (P_n, f_n), S)$$

*erzeugte **Sprache** ist*

$$\begin{aligned} L(\Gamma) \quad = \quad & \{w \in T^* \mid S \overset{f_{i_1}}{\Rightarrow}_{P_{i_1}} \alpha_1 \overset{f_{i_2}}{\Rightarrow}_{P_{i_2}} \ldots \overset{f_{i_r}}{\Rightarrow}_{P_{i_r}} \alpha_r = w, \\ & r \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq r\}. \end{aligned}$$

Wir nennen die Klasse aller hybriden CD Grammatiksysteme mit n Komponenten, die mit Ableitungsmodi $f_i \in M \subseteq \{t, \overline{t}, *, = k, \leq k, \geq k \mid k \geq 1\}$ arbeiten, $(HCD_n\text{-}CF, M)$. Die entsprechende Sprachklasse heißt $\mathcal{L}(HCD_n\text{-}CF, M)$. Wenn die Anzahl der Komponenten nicht beschränkt ist, wird n weggelassen.

Beispiel 5.1 Mit dem nun folgenden hybriden CD Grammatiksystem werden wir die nicht-kontextfreie Sprache $L_1 = \{a^m b^m c^n \mid 1 \leq m \leq n\}$ erzeugen.

$$\begin{aligned} \Gamma_1 \quad &= \quad (\{S, A, A', B, B', F\}, \{a, b, c\}, (P_1, t), (P_2, \overline{t}), (P_3, \overline{t}), S), \\ P_1 \quad &= \quad \{S \to AB, A \to aA'b, B \to B'c, A \to ab, B \to c\}, \\ P_2 \quad &= \quad \{B' \to B, A' \to F\}, \\ P_3 \quad &= \quad \{A' \to A, B \to F\}. \end{aligned}$$

Zuerst muß hier die Komponente P_1 angewendet werden. Sie führt die Satzform $aA'bB'c$ ein. Danach kann nur die Komponente P_2 aktiv werden. Nach den Vorschriften des $\overline{t}$-Modus, darf jetzt nur eine der beiden Produktionen angewendet werden. Mit $A' \to F$ wird die Ableitung blockiert, also kommt nur $B' \to B$ in Frage. Jetzt kann optional auf die gleiche Weise P_3 benutzt werden oder sofort wieder P_1 aktiv werden. Eine Ableitung sieht dann wie folgt aus:

$$\begin{aligned} S \quad & \overset{t}{\Longrightarrow}_{P_1} aA'bB'c \overset{\overline{t}}{\Longrightarrow}_{P_2} aA'bBc[\overset{\overline{t}}{\Longrightarrow}_{P_3} aAbBc] \\ & (\overset{t}{\Longrightarrow}_{P_1} \ldots \overset{\overline{t}}{\Longrightarrow}_{P_2} \ldots [\overset{\overline{t}}{\Longrightarrow}_{P_3}] \ldots)^{l-2} \overset{t}{\Longrightarrow}_{P_1} a^k A' b^k B' c^l \\ & \overset{\overline{t}}{\Longrightarrow}_{P_2} a^k A' b^k B c^l \overset{\overline{t}}{\Longrightarrow}_{P_3} a^k A b^k B c^l \overset{t}{\Longrightarrow}_{P_1} a^{k+1} b^{k+1} c^{l+1}. \end{aligned}$$

Dabei ist $1 \leq k \leq l$, da P_3 angewendet werden kann, aber nicht angewendet werden muß und deshalb gilt auch $L(\Gamma_1) = L_1$.

5.2 Generative Mächtigkeit

Die generative Mächtigkeit von hybriden CD Grammatiksystemen hängt stark von den benutzten Ableitungsmodi ab.

In [Mit93] finden sich die folgenden Ergebnisse für hybride CD Grammatiksysteme, die die klassischen Ableitungsmodi benutzen.

Satz 5.1 *Sei* $M_c = \{*, t, \leq k, = k, \geq k \mid k \geq 1\}$*, dann gilt*

$$\begin{aligned}\mathcal{L}(CF) &= \mathcal{L}(\text{HCD}_1\text{-CF}, M_c) \subset \mathcal{L}(\text{HCD}_2\text{-CF}, M_c) \subseteq \\ &\quad \mathcal{L}(\text{HCD}_3\text{-CF}, M_c) \subseteq \mathcal{L}(\text{HCD}_4\text{-CF}, M_c) = \mathcal{L}(\text{HCD-CF}, M_c).\end{aligned}$$

Außerdem wird in [Pău94] gezeigt, daß in $\mathcal{L}(\text{HCD-CF}, M_c)$ Sprachen enthalten sind, die nicht von ET0L-Systemen erzeugt werden können. Externe hybride CD Grammatiksysteme sind also mächtiger als CD Grammatiksysteme im t-Modus. Aus [BH00] sind die folgenden Ergebnisse über die generative Mächtigkeit des $\overline{t}$-Modus bekannt.

Satz 5.2 *Sei* $M_n = \{t, \overline{t}\}$*, dann gilt*

$$\mathcal{L}(\text{HCD-}CF, M_n) = \mathcal{L}(RP, ac).$$

Sei $M_m = \{\overline{t}, \geq k, = k, \leq k \mid k \geq 1\}$*, dann gilt*

$$\mathcal{L}(\text{HCD-}CF, M_m) = \mathcal{L}(\text{HCD}_4\text{-}CF, M_m) \subseteq \mathcal{L}(P).$$

Kombiniert man die Beweise für Satz 5.1 und den zweiten Punkt von 5.2, so erhält man

Korollar 5.1 *Sei* $M = \{*, t, \overline{t}, \leq k, = k, \geq k \mid k \geq 1\}$*, dann gilt*

$$\mathcal{L}(\text{HCD-}CF, M) = \mathcal{L}(\text{HCD}_5\text{-}CF, M).$$

Der in [Pău94] gezeigte Abschluß der Klasse $\mathcal{L}(\text{HCD-}CF, M_c)$ unter Vereinigung, Konkatenation, Schnitt mit regulären Mengen und Homomorphismus gilt mit den gleichen Argumenten auch für $\mathcal{L}(\text{HCD-}CF, M)$. Weitere Ergebnisse über Abschlußeigenschaften sind nicht bekannt.

5.3 Beschreibungskomplexität

Im letzten Abschnitt haben wir bereits gesehen, daß man die Anzahl der Komponenten Deg eines hybriden CD Grammatiksystems auf fünf beschränken kann.

Als nächstes wenden wir uns der maximalen Anzahl von Produktionen in einer Komponente und damit dem Maß $mProd$ zu. Auch für dieses Komplexitätsmaß gibt es eine konstante obere Schranke, wenn bestimmte Ableitungsmodi benutzt werden. Um die so erhaltenen Sprachklassen besser darstellen zu können, brauchen wir die folgende Definition.

Definition 5.4 *Die Klasse der hybriden CD Grammatiksysteme mit n Komponenten und höchstens m Produktionen in einer Komponente mit Ableitungsmodi aus $M \subseteq \{*, t, \overline{t}, \leq k, = k, \geq k \mid k \geq 1\}$ wird $(HCD_{n,m}\text{-}CF, M)$ genannt. Die Sprachklasse, die diese Klasse von hybriden CD Grammatiksystemen erzeugt, heißt dann $\mathcal{L}(HCD_{n,m}\text{-}CF, M)$. Ist n oder m nicht nach oben beschränkt schreiben wir an der entsprechenden Stelle ∞.*

Als erstes betrachten wir eine Kombination aus den Ableitungsmodi t und $(\geq k)$.

Lemma 5.1 *Sei $M = \{\geq k, t\}$, dann ist*

$$\mathcal{L}(HCD_{\infty,\infty}\text{-}CF, M) = \mathcal{L}(HCD_{\infty,6}\text{-}CF, M).$$

Beweis: Die Inklusion $\mathcal{L}(HCD_{\infty,6}\text{-}CF, M) \subseteq \mathcal{L}(HCD_{\infty,\infty}\text{-}CF, M)$ gilt nach der vorherigen Definition. Sei jetzt $L \in \mathcal{L}(HCD_{\infty,\infty}\text{-}CF, M)$. Dann gibt es $n, m > 0$ und ein hybrides CD Grammatiksystem

$$\Gamma = (N, T, (P_1, f_1), \ldots, (P_n, f_n), S)$$

in $(HCD_{n,m}\text{-}CF, M)$, das L erzeugt, wobei $f_1, \ldots, f_n \in M$ gilt. Wir konstruieren ein hybrides CD Grammatiksystem

$$\Gamma' = (N', T, P', S_0)$$

in $(HCD_{\infty,6}\text{-}CF, M)$, das die gleiche Sprache erzeugt. P' ist dabei eine Menge von Komponenten. Da die Konstruktion sehr viele Komponenten einführen wird und die Funktionsweise erst im Laufe des Beweises klar wird, werden sie hier noch nicht explizit aufgeführt.

Während des folgenden Beweises werden wir Γ als das originale Grammatiksystem und Γ' als das simulierende Grammatiksystem bezeichnen. Entsprechend nennen wir die Komponenten und Produktionen der beiden Grammatiksysteme.

Eine Satzform des simulierenden Grammatiksystems Γ' besteht aus vier Teilen. Diese Teile sind nicht unbedingt genau abgegrenzte Teilstrings der Satzform. Ein Nichtterminal bzw. Terminal in einer Satzform des simulierenden Grammatiksystems gehört aber logisch immer genau einem Teil an.

Der erste Teil enthält, abgesehen von ein paar technischen Details, die Satzform des originalen Grammatiksystems und ist auch der einzige Teil, der Terminale enthalten kann.

Der zweite Teil besteht aus dem Nichtterminal D, C oder F und zeigt den Status oder die Phase an, in dem sich das simulierende Grammatiksystem befindet.

Das Nichtterminal D zeigt dabei an, daß sich das simulierende Grammatiksystem in der Ableitungsphase (D-Phase) befindet. In dieser Phase werden die Ableitungsschritte des originalen Grammatiksystems simuliert. Das Nichtterminal C steht für die Wechselphase (C-Phase), in der geprüft wird, ob die Bedingungen des Ableitungsmodus der originalen Komponente eingehalten wurden (t-Modus oder ($\geq k$)-Modus). Danach wird die originale Komponente gewechselt. Das Nichtterminal F zeigt die Endphase (F-Phase) der Simulation an. In dieser Phase wird überprüft, ob die Satzform einen Terminalstring des originalen Grammatiksystems enthält. Danach werden die Nichtterminale, die zur Kontrolle der Ableitung benutzt wurden, gelöscht.

Der dritte Teil der simulierenden Satzform zeigt an, welche originale Komponente gerade aktiv ist und simuliert wird. Der vierte Teil der simulierenden Satzform schließlich enthält Nichtterminale, die zur Kontrolle der Ableitung benutzt werden.

Zu Beginn der Konstruktion von Γ' sind N' und P' leer. Als erstes wird zu N' die Menge N der originalen Nichtterminale hinzugefügt. Außerdem ergänzen wir N' mit der Menge $\{A' \mid A \in N\}$ und den neuen Nichtterminalen $S_0, C, D, F, Z, H, N_1, \ldots, N_n \notin N$. S_0 ist das neue Startsymbol.

Als nächstes beginnen wir, die Menge der Komponenten zu konstruieren. P' ist am Anfang der Konstruktion leer. Für den Beginn der Ableitung des simulierenden Grammatiksystems fügen wir folgende Produktionsmengen zu P' hinzu:

$$\begin{aligned}
(Q_s^1, t) &= \{S_0 \to SDN_2 \ldots N_n\}, \\
&\ldots \\
(Q_s^i, t) &= \{S_0 \to SDN_1 \ldots N_{i-1} N_{i+1} \ldots N_n\}, \\
&\ldots \\
(Q_s^n, t) &= \{S_0 \to SDN_1 \ldots N_{n-1}\}.
\end{aligned}$$

Die oben angegebenen Produktionen initialisieren die Satzform des simulierenden Grammatiksystems. Am Beginn jeder Ableitung muß genau eine Komponente Q_s^i für ein i mit $1 \leq i \leq n$ angewendet werden. In der resultierenden Satzform zeigt das Nichtterminal D an, daß sich Γ' in der Ableitungsphase befindet, und die Nichtterminale $N_1, \ldots, N_n$ zeigen an, welche Komponenten des originalen Grammatiksystems gerade nicht aktiv sind. Wenn also N_i nicht in der Satzform enthalten ist, wird die i-te Komponente des originalen Grammatiksystems gerade simuliert.

Als nächstes werden Komponenten definiert, die die Ableitungsschritte der originalen Komponenten, die im ($\geq k$)- oder im t-Modus arbeiten, simulieren. Die Simulation von Ableitungsschritten der originalen Komponenten findet im D-Status statt.

Für jede originale Komponente P_i mit $1 \leq i \leq n$ und $f_i = (\geq k)$ fügen wir ein neues Nichtterminal H_i zu N' hinzu. Außerdem ergänzen wir für jede dieser

Komponenten P_i und jedes $p \in P_i$, wobei $p : X \to \alpha$ ist, P' um die folgende Komponenten:

$$(Q_p^i, t) = \{X \to H_i\alpha, X \to X', C \to Z, F \to Z, N_i \to Z\}.$$

Für jede originale Komponente P_j, $1 \leq j \leq n$ mit $f_j = t$ und jedes $p \in P_j$, wobei $p : X \to \alpha$ ist, fügen wir zu P' die folgenden Komponenten hinzu:

$$(Q_p^j, t) = \{X \to \alpha, X \to X', C \to Z, F \to Z, N_j \to Z\}.$$

Die jeweils erste Produktion in den oben definierten Komponenten ist die originale Produktion. Im Fall des $(\geq k)$-Modus wird noch das Nichtterminal H_i hinzugefügt, das später benutzt wird, um die Gültigkeit der Ableitung zu überprüfen. Die jeweils zweite Produktion in den oben definierten Komponenten dient dazu, ein Nichtterminal X zu sperren. Da jede Komponente nur eine Produktion simuliert, können wir so erreichen, daß auch andere Produktionen, die das Nichtterminal X auf der linken Seite haben, bei der Simulation benutzt werden können. Die nächsten zwei Produktionen werden benutzt, um zu prüfen, ob die Ableitung im D-Status ist. Die Ableitung wird blockiert, falls das nicht der Fall ist. Die letzte Produktion beendet die Ableitung, falls die im Moment simulierte originale Komponente nicht P_i beziehungsweise P_j ist.

Um bei den Nichtterminalen X' die Markierung $'$ zu entfernen, wird für jedes $X \in N$ die Komponente

$$(Q_X, t) = \{C \to Z, F \to Z, X' \to X\}$$

zu P' hinzugefügt. Dabei überprüfen die ersten beiden Produktionen, ob die Satzform in der D-Phase ist. Ist das nicht der Fall, wird die Ableitung blockiert. Die letzte Produktion entfernt die Markierung $'$ von den Nichtterminalen X.

Die folgenden Komponenten werden benutzt, um zu prüfen, ob die Bedingungen des $(\geq k)$-Modus oder des t-Modus eingehalten wurden, während das simulierende Grammatiksystem die Ableitungsschritte einer Komponente des originalen Grammatiksystems simuliert hat.

Wenn wir die originalen $(\geq k)$-Komponenten P_i mit den neuen Komponenten Q_p^i simulieren, dann wird mit jeder Produktion $X \to H_i\alpha$ ein Nichtterminal H_i in die Satzform eingefügt. Um danach zu testen, ob auch k oder mehr Produktionen aus P_i simuliert wurden, werden die folgenden Komponenten zu P' hinzugefügt:

$$(Q_e^i, \geq k) = \{H_i \to \varepsilon\}.$$

Diese Komponenten können nur dann alle Nichtterminale H_i entfernen, wenn vorher k oder mehr Produktionen mit H_i auf der rechten Seite angewendet wurden. Sie sind auch die einzigen Komponenten, bei denen wir den $(\geq k)$-Modus brauchen.

Wir haben jetzt alle Komponenten definiert, die zur Simulation der Ableitungsschritte der originalen Komponenten nötig sind. Die neuen Komponenten sind so definiert, daß nur Komponenten, die zur aktuell simulierten originalen Komponente gehören, benutzt werden können, wenn die Ableitung in der D-Phase ist.

Als nächstes werden wir Komponenten definieren, mit denen man die originale Komponente wechseln kann. Mit der nächsten Komponente wechselt die Ableitung in die C-Phase. Für jede originale Komponente P_i, die im $(\geq k)$-Modus arbeitet, brauchen wir eine neue Komponente. Wir fügen C_i, $[C_1]$, ..., $[C_{n+1}]$ zu N' und die folgenden Komponenten zu P' hinzu:

$$(Q_c^i, t) = \{D \to CC_i[C_1] \dots [C_{n+1}], F \to Z, N_i \to Z, H_i \to Z\}.$$

Die erste Produktion wechselt den Status von D zu C und fügt einige Nichtterminale ein, die für spätere Gültigkeitsprüfungen benutzt werden. Mit C_i wird angezeigt, daß wir vorher die originale Komponente P_i simuliert haben. Mit der zweiten Produktion wird die Ableitung blockiert, wenn die Satzform im F-Status war. Die dritte Komponente beendet die Ableitung, falls die originale Komponente nicht P_i war. Und wenn Q_e^i die eingefügten Nichtterminale H_i vor der Anwendung von Q_c^i nicht löschen konnte, wird die Ableitung mit der vierten Produktion beendet. Arbeitet die originale Komponente im $(\geq k)$-Modus, kann die Ableitung des simulierenden Grammatiksystems nur weitergeführt werden, wenn mindestens k der originalen Produktionen angewendet wurden. Der $(\geq k)$-Modus wird also in Γ' korrekt simuliert.

Wir wenden uns jetzt wieder der Simulation der t-Modus Komponenten zu. Bevor wir von einer originalen Komponente P_j im t-Modus zu einer anderen originalen Komponente wechseln können, muß überprüft werden, ob die Vorschriften des t-Modus bei der Simulation auch eingehalten wurden. Seien $A_1, \dots, A_{n_j}$ die Nichtterminale in $dom(P_j)$. Wir fügen die neuen Nichtterminale $[A_1, j], \dots, [A_{n_j}, j]$ und $[A'_1, j], \dots, [A'_{n_j}, j]$ zu N' und die folgenden Komponenten zu P' hinzu:

$$\begin{aligned}(Q_c^j, t) \quad = \quad & \{D \to CC_j[C_1] \dots [C_{n+1}][A_1, j][A'_1, j] \dots [A_{n_j}, j][A'_{n_j}, j], \\ & F \to Z, N_j \to Z\}.\end{aligned}$$

Die erste Produktion wechselt den Status von D zu C und fügt einige Nichtterminale ein, die für spätere Gültigkeitsprüfungen benutzt werden. Das Nichtterminal C_j zeigt an, daß wir vorher die originale Komponente P_j simuliert haben. Mit der zweiten Produktion wird die Ableitung blockiert, wenn die Satzform im F-Status war. Die dritte Komponente beendet die Ableitung, falls die originale Komponente nicht P_j war.

Mit den neuen Nichtterminalen $[A_1, j], \dots, [A_{n_j}, j]$ und $[A'_1, j], \dots, [A'_{n_j}, j]$ können wir nach Anwendung von Q_c^j prüfen, ob die Vorschriften des t-Modus eingehalten wurden. Wenn ein Nichtterminal aus $dom(P_j)$ in der Satzform vorkommt, sind die Anforderungen des t-Modus nicht erfüllt und die Ableitung wird von den folgenden Komponenten, die wir für diese Prüfung einfügen, blockiert.

$$\begin{aligned}(Q_{c,1}^j, t) \quad &= \quad \{[A_1, j] \to \varepsilon, A_1 \to Z, D \to Z, F \to Z\}, \\ (Q_{c,2}^j, t) \quad &= \quad \{[A_2, j] \to \varepsilon, A_2 \to Z, [A_1, j] \to Z, D \to Z, F \to Z\}, \\ &\dots\end{aligned}$$

$$
\begin{aligned}
(Q^j_{c,n_j}, t) &= \{[A_{n_j}, j] \to \varepsilon, A_{n_j} \to Z, [A_{n_j-1}, j] \to Z, D \to Z, F \to Z\},\\
(Q^j_{c,n_j+1}, t) &= \{[A'_1, j] \to \varepsilon, A'_1 \to Z, [A_{n_j}, j] \to Z, D \to Z, F \to Z\},\\
(Q^j_{c,n_j+2}, t) &= \{[A'_2, j] \to \varepsilon, A'_2 \to Z, [A'_1, j] \to Z, D \to Z, F \to Z\},\\
&\dots\\
(Q^j_{c,2n_j}, t) &= \{[A'_{n_j}, j] \to \varepsilon, A'_{n_j} \to Z, [A'_{n_j-1}, j] \to Z, D \to Z, F \to Z\}.
\end{aligned}
$$

Bei allen eben angegebenen Komponenten wird durch die letzten beiden Produktionen sichergestellt, daß sich die Satzform in der C-Phase befindet. Durch die Produktionen $[A_p, j] \to \varepsilon$ und $A_p \to Z$ bzw. $[A'_p, j] \to \varepsilon$ und $A'_p \to Z$ mit $1 \leq p \leq n_j$ wird erzwungen, daß die Komponenten in der angegebenen Reihenfolge ausgeführt werden. Außerdem wird auf diese Art und Weise für jedes Nichtterminal A_p mit $1 \leq p \leq n_j$ überprüft, ob es in der Satzform vorhanden ist. Wenn eines dieser Nichtterminale gefunden wurde, sind die Anforderungen des t-Modus der originalen Komponente P_j nicht erfüllt und die Ableitung wird blockiert.

Als nächstes wenden wir uns den Schritten in der Simulation zu, in denen die aktuelle originale Komponente gewechselt wird. Die im folgenden definierten Komponenten werden dazu benutzt, C_i durch N_i zu ersetzen. Danach befindet sich die Satzform in einem Zustand, in der keine aktuelle zu simulierende Komponente angegeben ist.

$$
\begin{aligned}
(Q^1_{cn}, t) &= \{C_1 \to N_1, [A'_{n_1}, 1] \to Z, [C_1] \to \varepsilon, D \to Z, F \to Z\},\\
(Q^2_{cn}, t) &= \{C_2 \to N_2, [A'_{n_2}, 2] \to Z, [C_2] \to \varepsilon, [C_1] \to Z, D \to Z, F \to Z\},\\
&\dots\\
(Q^n_{cn}, t) &= \{C_n \to N_n, [A'_{n_n}, n] \to Z, [C_n] \to \varepsilon, [C_{n-1}] \to Z, D \to Z,\\
&\quad F \to Z\}.
\end{aligned}
$$

Wenn vorher die originale Komponente P_i war, ist ein Nichtterminal C_i in der Satzform vorhanden und wird mit der jeweils ersten Produktion in den oben stehenden Komponenten durch N_i ersetzt. Auf diese Weise befinden sich nach der Anwendung dieser Komponenten die Nichtterminale $N_1, \dots, N_n$ in der Satzform. Wenn die originale Komponente P_i im t-Modus gearbeitet hat, dann stellen wir mit den Produktionen $[A'_{n_i}, i] \to Z$ mit $A_{n_i} \in dom(P_i)$ sicher, daß alle Komponenten zur Prüfung der Vorschriften des t-Modus wie vorher definiert ausgeführt wurden. Ist dies nicht der Fall, wird die Ableitung blockiert. Wenn die originale Komponente aber im $(\geq k)$-Modus gearbeitet hat, ist kein Nichtterminal vom Typ $[A'_{n_i}, i]$ in der Satzform vorhanden. Deshalb wird die Ableitung auch dann nicht fälschlicherweise gestoppt. Die Produktionen $[C_i] \to \varepsilon$ und $[C_{i-1}] \to Z$ stellen sicher, daß alle eben definierten Komponenten in der angegebenen Reihenfolge ausgeführt werden. Mit den jeweils letzten beiden Produktionen wird dann sichergestellt, daß die Satzform in der C-Phase ist und andernfalls blockiert wird.

In Abbildung 5.1 wird die Simulation einer originalen $(= k)$-Modus Komponente mit allen simulierenden Komponenten dargestellt. Dabei ist wichtig, daß

die Nichtterminale in der Abbildung nach den am Anfang des Beweises angegebenen Teilen sortiert sind, was in der simulierenden Satzform nicht der Fall ist und nur wegen der besseren Übersichtlichkeit so dargestellt wurde.

In Abbildung 5.2 sieht man die Simulation einer originalen t-Modus Komponente mit allen simulierenden Komponenten. Die Anordnung der Nichtterminale ist auch hier wie bei der vorherigen Abbildung vereinfacht.

Um zur Simulation der nächsten Komponente zu wechseln, muß ein N_j mit $1 \leq j \leq n$ gelöscht werden. Die aktuell zu simulierende Komponente ist dann die originale Komponente P_j. Mit einer der n folgenden Komponenten wählen wir die originale Komponente aus, die als nächstes simuliert werden soll. Wir brauchen dazu neue Nichtterminale $B_1, \ldots, B_n \notin N$, die wir zu N' hinzufügen. Darüber hinaus fügen wir zu P' neue Komponenten hinzu, die wie folgt definiert sind:

$$\begin{aligned}
(Q_b^1, t) &= \{N_1 \to B_2 \ldots B_n, B_1 \to Z, [C_n] \to Z, [C_{n+1}] \to \varepsilon, \\
&\quad D \to Z, F \to Z\}, \\
&\ldots \\
(Q_b^n, t) &= \{N_n \to B_1 \ldots B_{n-1}, B_n \to Z, [C_n] \to Z, [C_{n+1}] \to \varepsilon, \\
&\quad D \to Z, F \to Z\}.
\end{aligned}$$

Es sei bemerkt, daß nur eine der oben definierten Komponenten angewendet werden kann, bevor wir im nächsten Schritt wieder in den D-Status oder zum F-Status wechseln. Die Satzform bleibt also gültig, d.h. sie behält die Form, die wir am Anfang angegeben haben. Das wird durch die jeweils ersten beiden Produktionen erreicht. Die erste Produktion wählt die neue originale Komponente P_j aus, indem sie N_j durch $B_1 \ldots B_{j-1} B_{j+1} \ldots B_n$ ersetzt. Mit der jeweils zweiten Produktion wird die Ableitung dann blockiert, wenn eine weitere Komponente $Q_b^{j'}$ mit $1 \leq j' \neq j \leq n$ angewendet wird. Mit $[C_n] \to Z$, stellen wir sicher, daß die vorher definierten Komponenten $Q_{cn}^1, \ldots, Q_{cn}^n$ angewendet wurden, und die letzten beiden Produktionen blockieren die Ableitung, wenn sich die Satzform nicht in der C-Phase befindet.

Die simulierende Satzform besteht jetzt aus einer Satzform, die vom originalen CD Grammatiksystem erzeugt werden kann. Außerdem enthält sie Nichtterminale zur Steuerung wie am Anfang des Beweises beschrieben. Die Simulation einer weiteren Komponente kann also beginnen.

Mit den als nächstes definierten Komponenten wechselt die Satzform vom C- zum D-Status. Außerdem werden die mit den vorher definierten Komponenten eingeführten B_is gelöscht.

$$\begin{aligned}
(Q_{cd}^1, t) &= \{[C_{n+1}] \to Z, C \to D, B_1 \to \varepsilon\}, \\
&\ldots \\
(Q_{cd}^n, t) &= \{[C_{n+1}] \to Z, C \to D, B_n \to \varepsilon\}.
\end{aligned}$$

Die Komponenten $Q_{cd}^1, \ldots, Q_{cd}^n$ müssen alle benutzt werden, weil sonst B_is in der Satzform bleiben, die später die Ableitung blockieren würden. Außerdem

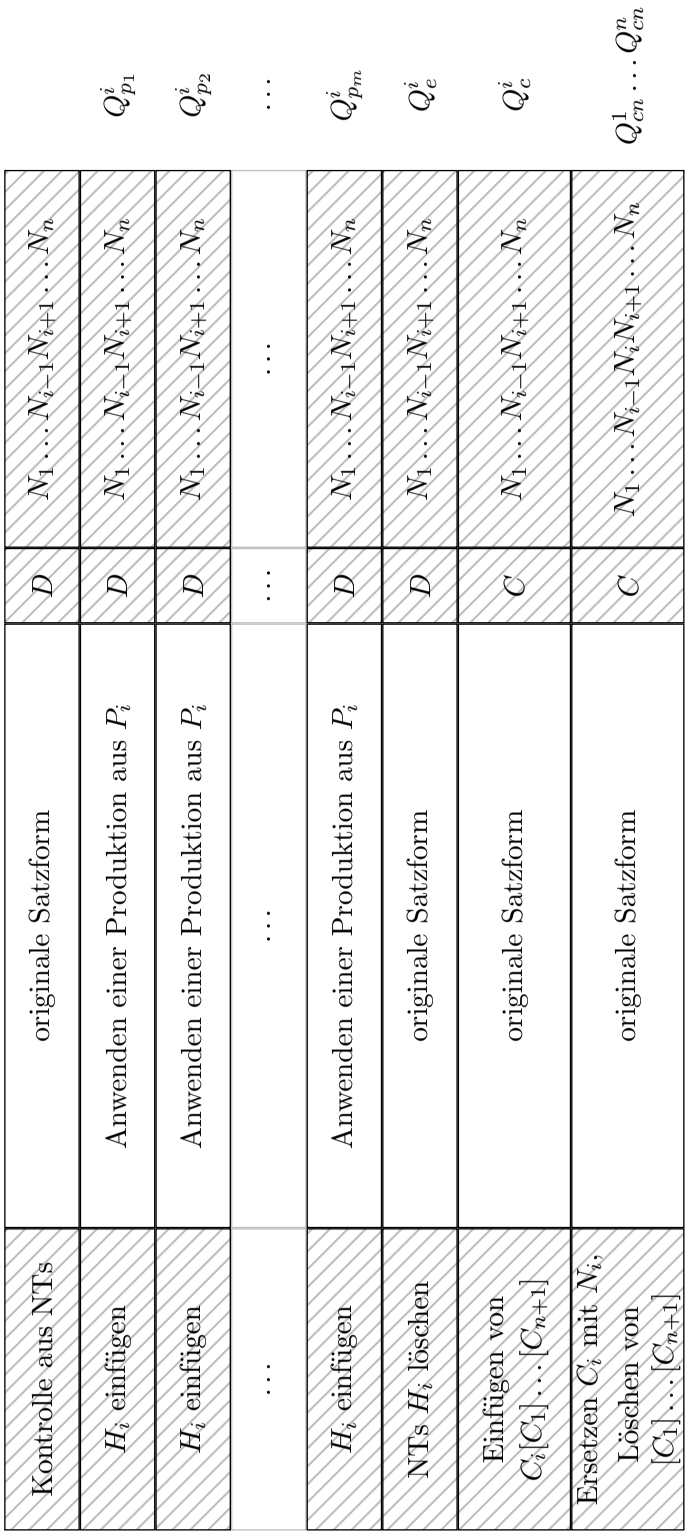

Abbildung 5.1: Simulation einer $(=k)$-Modus Komponente P_i

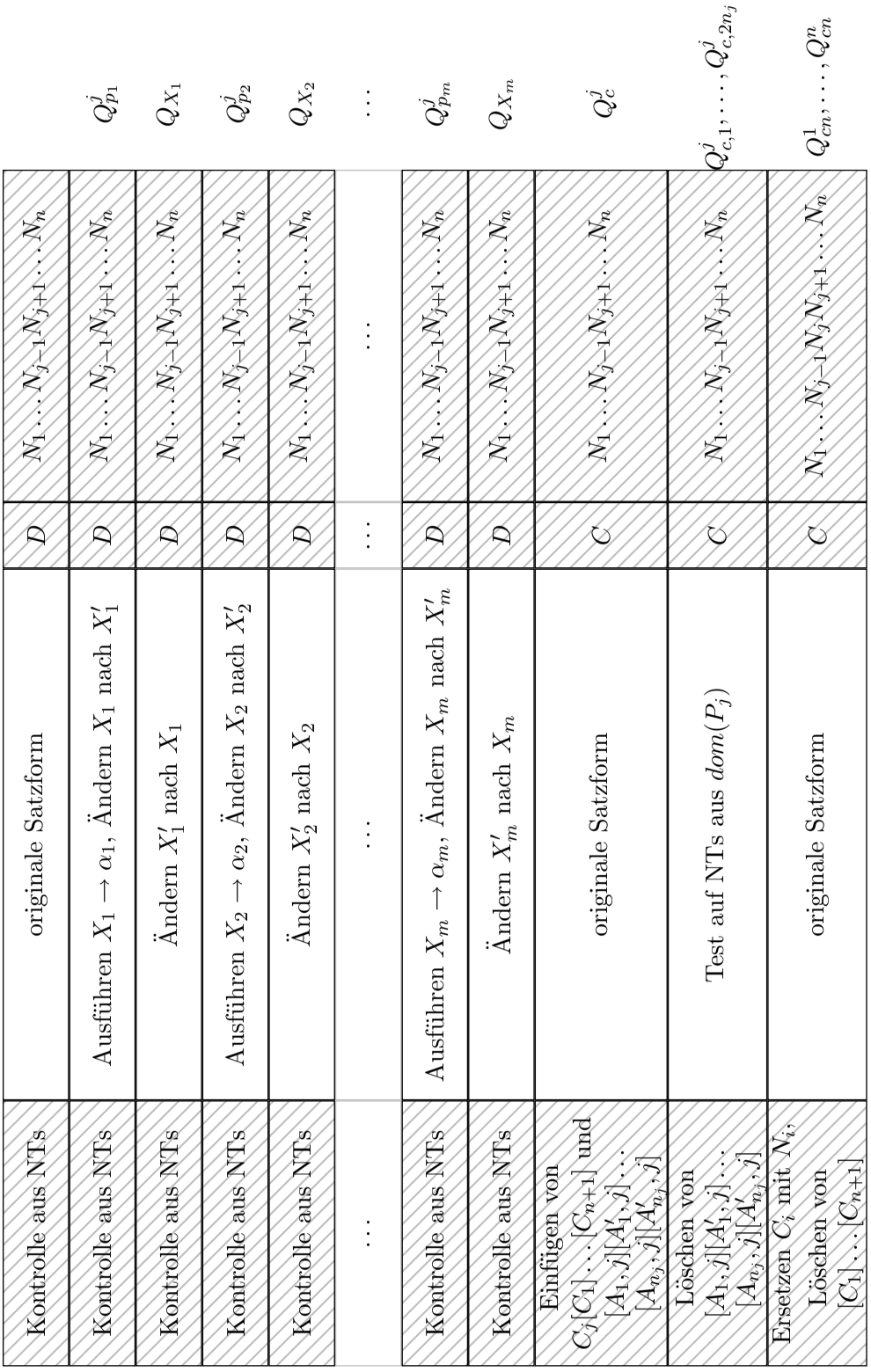

Abbildung 5.2: Simulation einer t-Modus Komponente P_j

wird mit der jeweils ersten Produktion sichergestellt, daß eine der Komponenten $Q_b^1, \ldots, Q_b^n$ benutzt wurde, sonst würde ein Nichtterminal $[C_{n+1}]$ in der Satzform stehen und die Ableitung beendet werden. Die zweite Produktion wechselt die Phase von C wieder auf D. Insgesamt bleibt die Satzform also in der Form, die am Anfang des Beweises angegeben wurde, während die gerade simulierte originale Komponente geändert wird und von Phase C auf Phase D gewechselt wird.

In Abbildung 5.3 sieht man, wie die simulierende Satzform vom C- zum D-Status wechselt und dabei eine neue originale Komponente ausgewählt wird. Die Anordnung der Nichtterminale ist auch hier wie bei den vorherigen Abbildungen vereinfacht.

Nachdem die Ableitung des originalen CD Grammatiksystems simuliert wurde, wechselt die simulierende Ableitung in den F-Status. Im F-Status werden die Nichtterminale, die zur Kontrolle der simulierenden Ableitung benutzt wurden, entfernt. Davor überprüfen die eingefügten Komponenten die Ableitung auf ihre Gültigkeit.

Seien $A_1, \ldots, A_t$ alle Nichtterminale von Γ. Wir fügen die neuen Nichtterminale $[A_1], \ldots, [A_t]$ und $[N_1], \ldots, [N_n]$ zu N' hinzu. Außerdem werden die folgenden Komponenten zu P' hinzugefügt. Sie dienen dazu, die simulierende Satzform in den F-Status zu bringen und zu testen, ob die originale Satzform einen Terminalstring enthält.

$$\begin{aligned}
(Q_{cf}^1, t) &= \{[C_{n+1}] \to Z, C \to F[A_1] \ldots [A_t][N_1] \ldots [N_n], B_1 \to \varepsilon\}, \\
&\ldots \\
(Q_{cf}^n, t) &= \{[C_{n+1}] \to Z, C \to F[A_1] \ldots [A_t][N_1] \ldots [N_n], B_n \to \varepsilon\}.
\end{aligned}$$

Die Komponenten $Q_{cf}^1, \ldots, Q_{cf}^n$ müssen alle benutzt werden, weil sonst B_is in der Satzform bleiben, die später die Ableitung blockieren würden. Außerdem wird mit der jeweils ersten Produktion sichergestellt, daß eine der Komponenten $Q_b^1, \ldots, Q_b^n$ benutzt wurde, sonst würde ein Nichtterminal $[C_{n+1}]$ in der Satzform stehen und die Ableitung beendet werden. Mit der jeweils zweiten Produktion wird in den F-Status gewechselt. Darüber hinaus werden die eben definierten Nichtterminale für die Gültigkeitsprüfungen in die Satzform eingefügt.

Um festzustellen, ob kein Nichtterminal der originalen Grammatik mehr in der Satzform ist, fügen wir die folgenden Komponenten zu P' hinzu.

$$\begin{aligned}
(Q_{f,1}, t) &= \{D \to Z, C \to Z, [A_1] \to \varepsilon, A_1 \to Z, A_1' \to Z\}, \\
(Q_{f,2}, t) &= \{D \to Z, C \to Z, [A_1] \to Z, [A_2] \to \varepsilon, A_2 \to Z, A_2' \to Z\}, \\
&\ldots \\
(Q_{f,t}, t) &= \{D \to Z, C \to Z, [A_{t-1}] \to Z, [A_t] \to \varepsilon, A_t \to Z, A_t' \to Z\}.
\end{aligned}$$

Die jeweils ersten beiden Produktionen aus diesen Komponenten blockieren die Ableitung, wenn die Satzform nicht im F-Status ist. Mit den Produktionen

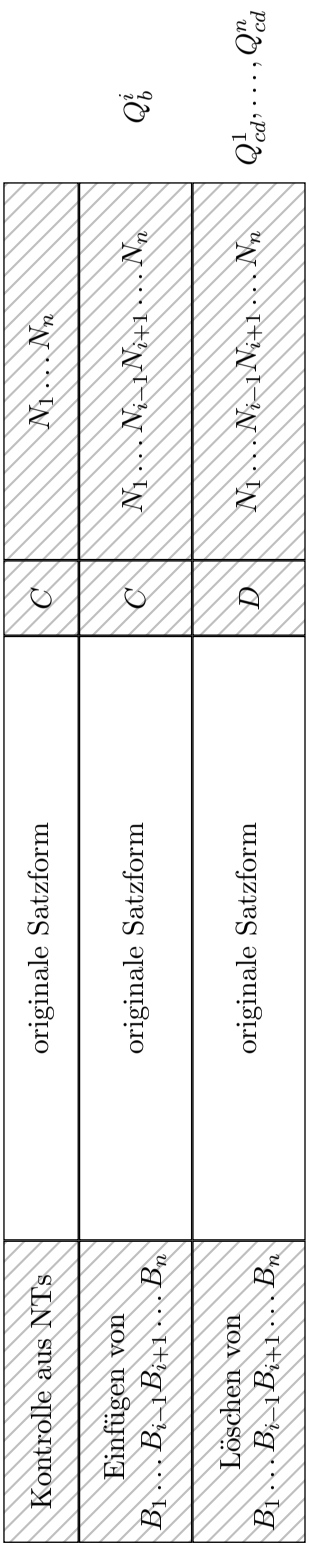

Abbildung 5.3: Wechsel der simulierenden Satzform vom C- zum D-Status und Auswahl der originalen Komponente P_i

$A_i \to Z$ mit $1 \leq i \leq t$ wird die Ableitung beendet, falls ein Nichtterminal aus dem originalen CD Grammatiksystem vorhanden ist. Außerdem sorgen die Produktionen der Form $[A_i] \to Z$ und $[A_{i+1}] \to \varepsilon$ dafür, daß alle eben definierten Komponenten in der vorgegebenen Reihenfolge ausgeführt werden.

Als nächstes müssen die Nichtterminale, die die originale Komponente und den Status der Ableitung anzeigen, gelöscht werden. Dafür werden die folgenden Komponenten zu P' hinzugefügt.

$$\begin{aligned}
(Q_{t,1}, t) &= \{D \to Z, C \to Z, [A_t] \to Z, [N_1] \to \varepsilon, N_1 \to \varepsilon\}, \\
(Q_{t,2}, t) &= \{D \to Z, C \to Z, [N_1] \to Z, [N_2] \to \varepsilon, N_2 \to \varepsilon\}, \\
&\dots \\
(Q_{t,n}, t) &= \{D \to Z, C \to Z, [N_{n-1}] \to Z, [N_n] \to \varepsilon, N_n \to \varepsilon\}, \\
(Q_{t,n+1}, t) &= \{D \to Z, C \to Z, [N_n] \to Z, F \to \varepsilon\}.
\end{aligned}$$

Die jeweils ersten beiden Produktionen aus diesen Komponenten blockieren die Ableitung, wenn die Satzform nicht im F-Status ist. Die Produktion $[A_t] \to Z$ in $Q_{t,1}$ stellt sicher, daß vorher $Q_{f,1}, \dots, Q_{f,t}$ angewendet wurde und durch die Produktionen der Form $[N_i] \to Z$ und $[N_{i+1}] \to \varepsilon$ wird erzwungen, daß $Q_{t,1}, \dots, Q_{t,n+1}$ in der angegebenen Reihenfolge ausgeführt werden. Die jeweils letzte Produktion schließlich löscht die Nichtterminale zur Steuerung der Ableitung $N_1, \dots, N_n, F$.

Es sei angemerkt, daß sowohl $Q_{f,1}, \dots, Q_{f,t}$ als auch $Q_{t,1}, \dots, Q_{t,n+1}$ angewendet werden müssen, um einen Terminalstring zu erhalten.

In Abbildung 5.4 sieht man, wie die simulierende Satzform vom C- zum F-Status überführt wird und danach alle Nichtterminale, die zur Kontrolle der Ableitung dienen, aus der Satzform entfernt werden. Auch hier ist die Anordnung der Nichtterminale wie bei den vorherigen Abbildungen vereinfacht.

Durch die Konstruktion der Komponenten und die Tatsache, daß alle Produktionen aus den Komponenten von Γ verwendet wurden, ist gewährleistet, daß $L(\Gamma, M) \subset L(\Gamma', M)$ gilt.

Die Produktionen im F-Status können nur in der vorgegebenen Reihenfolge benutzt werden. Wenn der F-Status einmal erreicht wurde, kann nicht mehr in einen anderen Status gewechselt werden. Nur wenn alle Komponenten des F-Status benutzt wurden, kann ein Terminalstring erreicht werden. Andererseits gibt es keine Möglichkeit, eine Komponente aus dem F-Status in einem anderen Status zu benutzen.

Auch die Komponenten, die zum C- oder D-Status gehören, können nur im C- beziehungsweise D-Status benutzt werden. Die einzigen Komponenten, die im „falschen“ Status benutzt werden können, sind die Komponenten, die vom C- zum D- oder F-Status wechseln. Sie verändern allerdings die gültige Form der Ableitung nicht, wenn sie an der falschen Stelle angewendet werden, und führen deshalb auch nicht zu einem Terminalstring, der nicht in der Sprache des originalen CD Grammatiksystems ist.

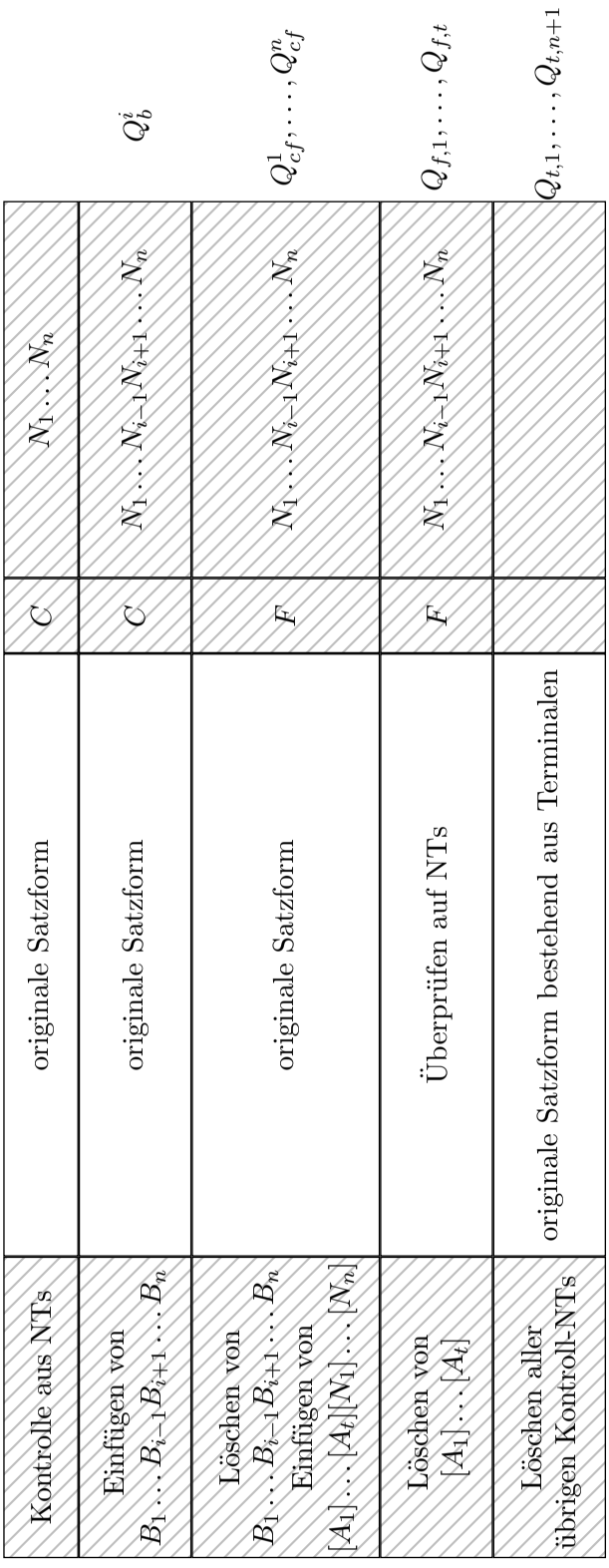

Abbildung 5.4: Wechsel der simulierenden Satzform vom C- zum F-Status und Entfernen der Kontroll-Nichtterminale

Wenn jetzt also ein Wort $w \in L(\Gamma', M)$ ist, dann gibt es eine Reihenfolge von Komponenten aus Γ, die simuliert wurden, um w zu erzeugen. Die Ableitung eines Wortes aus Terminalsymbolen kann nur erfolgen, wenn die Komponenten korrekt simuliert werden und auch der Wechsel zwischen den Komponenten korrekt erfolgt, sonst wird die Ableitung in Γ' blockiert. Ist die Ableitung einmal in die F-Phase übergegangen, kann diese nicht mehr verlassen werden. Es gibt jetzt die Möglichkeit, daß in der Ableitung vorher ein Terminalwort aus $L(\Gamma, M)$ erzeugt wurde und nur noch Hilfsnichtterminale aus Γ' in der Satzform vorhanden sind, die in der F-Phase gelöscht werden. Dann gilt auch $w \in L(\Gamma, M)$. Andernfalls befinden sich noch Nichtterminale aus Γ oder Hilfsnichtterminale aus Γ', die nicht in der F-Phase gelöscht werden, in der Satzform. Die Ableitung wird dann blockiert. Insgesamt gilt auch $L(\Gamma', M) \subseteq L(\Gamma, M)$. □

Lemma 5.2 *Wenn $M = \{=k, t\}$, dann gilt*

$$\mathcal{L}_M(HCD_{\infty,\infty}CF) = \mathcal{L}_M(HCD_{\infty,6}CF).$$

Beweis: Der Beweis stimmt fast völlig mit dem von Lemma 5.1 überein. Der einzige Unterschied ist, daß die $(\geq k)$-Komponenten in Lemma 5.1 durch $(=k)$-Komponenten der folgenden Form ersetzt werden müssen.

$$(Q_e^i, =k) = \{H_i \to \varepsilon\}.$$

□

Lemma 5.3 *Wenn $M = \{\leq k, t\}$, dann gilt*

$$\mathcal{L}_M(HCD_{\infty,\infty}CF) = \mathcal{L}_M(HCD_{\infty,6}CF).$$

Beweis: Der Beweis stimmt fast völlig mit dem von Lemma 5.1 überein. Der einzige Unterschied ist, daß die $(\geq k)$-Komponenten aus Lemma 5.1 durch die $(\leq k)$-Komponenten

$$(Q_e^i, \leq k) = \{H_i \to \varepsilon\}$$

ersetzt werden müssen. □

Lemma 5.4 *Wenn $M = \{\overline{t}, t\}$, dann gilt*

$$\mathcal{L}(HCD_{\infty,\infty}\text{-}CF, M) = \mathcal{L}(HCD_{\infty,6}\text{-}CF, M).$$

Beweis: Der Beweis ist ähnlich wie der von Lemma 5.1. Anstelle der Komponenten, die im Beweis von Lemma 5.1 für die $(\geq k)$-Komponenten in Γ konstruiert wurden, müssen für die $\overline{t}$-Komponenten andere simulierende Komponenten zu P' hinzugefügt werden.

Für jede originale Komponente P_i mit $1 \leq i \leq n$ und $f_i = \overline{t}$ ergänzen wir für jedes $p \in P_i$, wobei $p : X \to \alpha$ ist, P' um die folgende Komponente:

$$(Q_p^i, t) = \{X \to \alpha, X \to X', C \to Z, F \to Z, N_i \to Z\}.$$

Wenn die originale Komponente P_i im t-Modus arbeitet, ist die Simulation der Ableitungsschritte, der Test, ob die Vorschriften des t-Modus eingehalten wurden und der Wechsel vom D- zum C-Status genau wie in den vorherigen Beweisen definiert.

Wenn die originale Komponente P_i im $\overline{t}$-Modus arbeitet, fügen wir ein neues Nichtterminal K_i zu N' hinzu. Mit der folgenden Komponente wird zum C-Status gewechselt.

$$(Q_c^i, t) = \{D \to CK_iC_i[C_1] \dots [C_{n+1}], F \to Z, N_i \to Z\}.$$

Danach muß überprüft werden, ob die Vorschriften des $\overline{t}$-Modus während der Simulation einer $\overline{t}$-Komponente eingehalten wurden. Dazu wird festgestellt, ob ein Nichtterminal aus $dom(P_i)$ in der Satzform vorhanden ist. Ist das nicht der Fall, wird die Ableitung blockiert.

Seien $A_1, \dots, A_{n_i}$ die Nichtterminale aus $dom(P_i)$. Für jede Komponente P_i, die im $\overline{t}$-Modus arbeitet, fügen wir die folgenden Komponenten zu P' hinzu:

$$\begin{aligned}
(Q_{c,1}^i, \overline{t}) &= \{A_1 \to A_1, A_1' \to A_1, K_i \to \varepsilon\}, \\
&\dots \\
(Q_{c,n_i}^i, \overline{t}) &= \{A_{n_i} \to A_{n_i}, A_{n_i}' \to A_{i,n_i}, K_i \to \varepsilon\}.
\end{aligned}$$

Durch die Vorgaben des $\overline{t}$-Modus kann das Nichtterminal K_i nur gelöscht werden, wenn mindestens eines der Nichtterminale aus $dom(P_i)$ in der Satzform vorhanden ist, also eine der eben definierten Komponenten im $\overline{t}$-Modus angewendet werden kann.

Als nächstes definieren wir die Komponenten, die dazu benutzt werden, C_i durch N_i zu ersetzen.

$$\begin{aligned}
(Q_c^1, t) &= \{C_1 \to N_1, K_1 \to Z, [A_{n_1}', 1] \to Z, [C_1] \to \varepsilon\}, \\
(Q_c^2, t) &= \{C_2 \to N_2, K_2 \to Z, [A_{n_2}', 2] \to Z, [C_2] \to \varepsilon, [C_1] \to Z\}, \\
&\dots \\
(Q_c^n, t) &= \{C_n \to N_n, K_n \to Z, [A_{n_n}', n] \to Z, [C_n] \to \varepsilon, [C_{n-1}] \to Z\}.
\end{aligned}$$

Diese Komponenten ersetzen $Q_c^1, \dots, Q_c^n$ aus Lemma 5.1. Die Veränderungen in den Komponenten betreffen aber nicht die Prüfung darauf, ob die t-Modus Voraussetzungen eingehalten wurden. Wenn allerdings K_i vorher nicht gelöscht werden konnte, wird die Ableitung blockiert. Das passiert nur, wenn eine Komponente P_i im $\overline{t}$-Modus nicht korrekt simuliert wurde. Die Komponenten, die im originalen CD Grammatiksystem im $\overline{t}$-Modus gearbeitet haben, werden also vom neuen CD Grammatiksystem korrekt simuliert.

In Abbildung 5.5 sieht man die Simulation einer originalen $\overline{t}$-Modus Komponente mit allen simulierenden Komponenten. Die Anordnung der Nichtterminale ist auch hier wie bei der vorherigen Abbildung vereinfacht. □

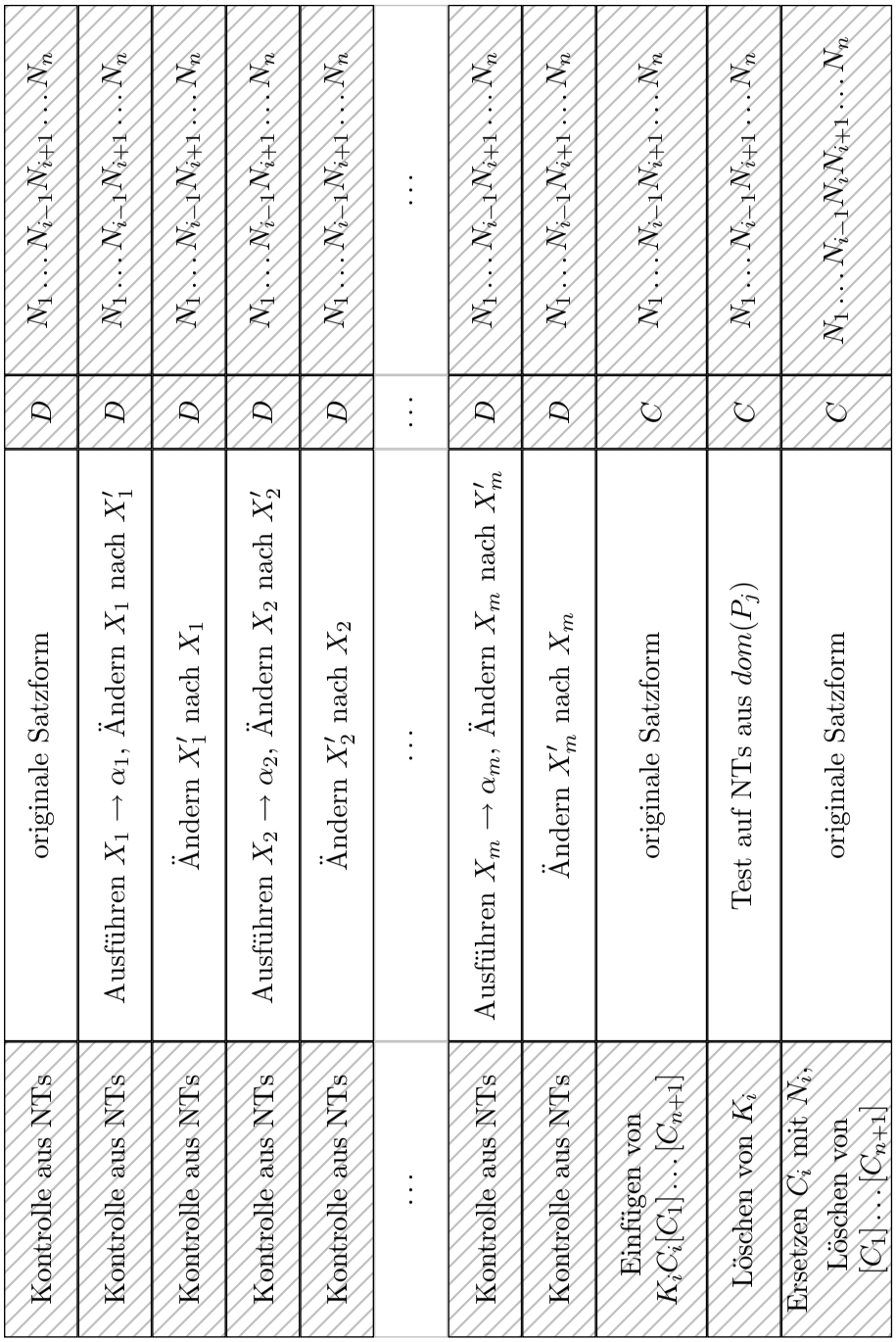

Abbildung 5.5: Simulation einer $\overline{t}$-Modus Komponente P_i

Durch eine Kombination der vorangegangenen Lemmata erhalten wir den folgenden allgemeineren Satz:

Satz 5.3 *Wenn $M \subseteq \{*, t, \overline{t}, \leq k, = k, \geq k \mid k \geq 1\}$ und $t \in M$ ist, dann gilt*

$$\mathcal{L}(HCD_{\infty,\infty}\text{-}CF, M) = \mathcal{L}(HCD_{\infty,6}\text{-}CF, M).$$

Wie man an den bisherigen Ergebnissen dieses Abschnitts sehen kann, ist es möglich, die Anzahl der Komponenten beziehungsweise die maximale Anzahl von Produktionen in einer Komponente zu beschränken. Allerdings erhöhen die angegebenen Konstruktionen den jeweils anderen Parameter. Wenn wir jetzt also Deg oder $mProd$ festhalten, stellt sich die Frage, wie sich die generative Mächtigkeit bzw. die Beschreibungskomplexität des Grammatiksystems bei steigender Größe des anderen Parameters verhalten. Ein ähnliches Resultat aus [PDS94] für kontextfreie CD Grammatiksysteme gilt entsprechend modifiziert auch für extern hybride CD Grammatiksysteme.

Lemma 5.5 *Wenn $M \subseteq \{*, t, \overline{t}, \leq k, = k, \geq k \mid k \geq 1\}$ ist, dann gilt für alle $n, m \geq 1$*

1. $\mathcal{L}(HCD_{n,m}\text{-}CF, M) \subset \mathcal{L}(HCD_{n+1,m}\text{-}CF, M)$,
2. $\mathcal{L}(HCD_{n,m}\text{-}CF, M) \subset \mathcal{L}(HCD_{n,m+1}\text{-}CF, M)$.

Beweis: Sei

$$L_{m(n+1)} = \{a_1, a_2, \ldots, a_{m(n+1)}\}.$$

Wenn $M \subseteq \{*, t, \leq k, = 1, \geq 1 \mid k \geq 1\}$ ist, dann wird $L_{m(n+1)}$ von dem externen hybriden CD Grammatiksystem

$$\Gamma = (\{S\}, T, (P_1, f_1), \ldots, (P_{n+1}, f_{n+1}), S)$$

mit $f_1, \ldots, f_{n+1} \in M$ und

$$P_i = \{S \to a_{mi}, S \to a_{mi-1}, \ldots, S \to a_{mi-(m-1)}\},$$

für alle $1 \leq i \leq n+1$ erzeugt. Wir erhalten $L_{m(n+1)} \in \mathcal{L}(HCD_{n+1,m}\text{-}CF, M)$.

Andererseits braucht man für die Erzeugung von a_i mit $1 \leq i \leq (n+1)m$, jeweils eine Regel mit der rechten Seite a_i, aber ein hybrides CD Grammatiksystem in $HCD_{n,m}$-CF hat höchstens nm Regeln. Daraus folgt, daß $L_{m(n+1)} \notin \mathcal{L}(HCD_{n,m}\text{-}CF, M)$.

Wenn $M \subseteq \{*, t, \leq k, = k', \geq k' \mid k \geq 1, k' \geq 2\}$ ist, dann braucht jede $(= k')$- oder $(\geq k')$-Modus Ableitung eine Produktion mehr, zum Beispiel $S \to S$. Sei n' die Anzahl der Ableitungsmodi aus $\{= k', \geq k' \mid k' \geq 2\}$ in M, dann erzeugt das folgende extern hybride CD Grammatiksystem

$$\Gamma = (\{S\}, T, (P_1, f_1), \ldots, (P_{n+1}, f_{n+1}), S)$$

die Sprache $L_{m(n+1)-n'}$. Die Modi $f_1, \ldots, f_{n-n'}$ sind aus $\{*, t, \leq k, = 1, \geq 1 \mid k \geq 1\}$ und

$$P_i = \{S \to a_{mi}, S \to a_{mi-1}, \ldots, S \to a_{mi-(m-1)}\},\ 1 \leq i \leq n - n'.$$

Die Modi $f_{n-n'+1}, \ldots, f_n$ sind in $\{=k', \geq k' \mid k' \geq 2\}$ und die Komponenten P_j enthalten jeweils eine nichtterminierende Produktion:

$$\begin{aligned} P_j \quad = \quad & \{S \to S, S \to a_{m(n-n')+j(m-1)}, S \to a_{m(n-n')+j(m-1)-1}, \ldots, \\ & S \to a_{m(n-n')+j(m-1)-(m-2)}\}, \; n - n' < j \leq n. \end{aligned}$$

Mit der minimalen Anzahl von $\{= k', \geq k' \mid k' \geq 2\}$-Komponenten erhalten wir $L_{m(n+1)-n'} \in \mathcal{L}(HCD_{n+1,m}\text{-}CF, M)$.

Um die Sprache $L_{m(n+1)-n'}$ zu erhalten, brauchen wir mindestens $m(n+1)-n'$ terminierende Produktionen. Wir brauchen aber auch nichtterminierende Produktionen und zwar mindestens eine für jede Komponente im $\{= k', \geq k' \mid k' \geq 2\}$-Ableitungsmodus, also mindestens n' weitere Produktionen. Wenn wir diese nichtterminierenden Produktionen nicht haben, können die terminierenden Produktionen in diesen Komponenten nicht benutzt werden und können deshalb nicht die einzigen der Form $S \to a_i$ sein. Wir erhalten also $L_{m(n+1)-n'} \notin \mathcal{L}(HCD_{n,m}\text{-}CF, M)$.

Wenn $M \subseteq \{*, t, \overline{t}, \leq k, = k', \geq k' \mid k \geq 1, k' \geq 2\}$ ist, können wir ähnliche Argumente wie oben benutzen. Außerdem ist noch eine zusätzliche $\overline{t}$-Modus Komponente in jedem extern hybriden CD Grammatiksystem vorhanden, das $L_{m(n+1)-n'}$ erzeugt. Wir können aber keine andere Komponente entfernen, da eine $\overline{t}$-Komponente durch die Einschränkungen des $\overline{t}$-Modus Produktionen mit rechter Seite a_i nicht benutzen kann.

Für die unendliche Hierarchie über $mProd$ benutzen wir genau die gleichen Argumente wie in dem Beweis für die Hierarchie über Deg. □

5.4 Zusammenfassung

In diesem Kapitel haben wir hybride CD Grammatiksysteme untersucht. Dabei kam zu den schon vorher eingeführten Ableitungsmodi noch der $\overline{t}$-Modus hinzu. Im Abschnitt über die generative Mächtigkeit haben wir herausgefunden, daß man mit den klassischen Modi $\{t, *, = k, \geq k, \leq k \mid k \geq 1\}$ eine Klasse von Sprachen erzeugen kann, die mächtiger ist als die von kontextfreien CD Grammatiksystemen im t-Modus erzeugten ET0L Sprachen. Außerdem kann man, wenn man den $\overline{t}$ Modus benutzt, die „recurrent“ programmierten Grammatiken beziehungsweise eine Unterklasse der programmierten Grammatiken erreichen.

Ein weiteres wichtiges Ergebnis, das so nicht zu erwarten war, ist, daß man im Fall der klassischen Modi nur vier Komponenten braucht, um alle erreichbaren Sprachen zu erzeugen. Wenn man den $\overline{t}$-Modus hinzu nimmt, benötigt man nicht mehr als fünf Komponenten.

Im Bereich der Beschreibungskomplexität haben wir $mProd$, die maximale Anzahl von Produktionen in einer Komponente, untersucht. Dieses Maß konnten wir nach oben beschränken und zeigen, daß sechs Produktionen in einer

Komponente ausreichen, um alle Sprachen in der betrachteten Sprachklasse zu erzeugen. Wir konnten darüber hinaus feststellen, daß eine unendliche Hierarchie über dem Parameter $mProd$ oder Deg entsteht, wenn man den jeweils anderen Parameter festhält.

Kapitel 6

Metalineare Grammatiksysteme

Die Klasse der CD Grammatiksysteme und auch die der hybriden CD Grammatiksysteme erfüllt die Forderung nach Mechanismen zur Erzeugung nichtkontextfreier Sprachen mit kontextfreien Produktionen. Allerdings sind einige Fragen im Bereich der generativen Mächtigkeit noch nicht gelöst und auch wichtige Methoden zur Lösung dieser Fragen wie zum Beispiel Pumpinglemmata noch nicht gefunden. Auf dem Gebiet der Beschreibungskomplexität gibt es hauptsächlich Ergebnisse über Maße, die nur einzelne Aspekte eines CD Grammatiksystems widerspiegeln und nicht die gesamte Größe repräsentieren. Als nächstes werden wir deshalb die Form der Produktionen einschränken, um Sprachklassen zu erhalten, deren Handhabung einfacher ist.

Als erstes kämen dafür CD Grammatiksysteme mit regulären Komponenten in Frage. In [Mit93] wir allerdings gezeigt, daß diese nur die regulären Sprachen erzeugen. Dies ist auch bei hybriden CD Grammatiksystemen der Fall. Wir werden sehen, daß auch CD Grammatiksysteme mit linearen Komponenten nur lineare Sprachen erzeugen. Deshalb konzentrieren wir uns in diesem Kapitel auf die metalinearen CD Grammatiksysteme.

Nach benötigten Definitionen und einigen Beispielen werden im folgenden die wichtigsten Ergebnisse über die generative Mächtigkeit und Abschlußeigenschaften von metalinearen CD Grammatiksystemen präsentiert. Danach beschäftigen wir uns ausführlich mit der Beschreibungskomplexität von metalinearen CD Grammatiksystemen.

Die hier präsentierten Ergebnisse über die generative Mächtigkeit von CD Grammatiksystemen sind zum großen Teil in [Sun07] gezeigt worden. Für die Resultate über die Beschreibungskomplexität verweisen wir auf [Sun05b].

6.1 Definition und Beispiele

Definition 6.1 *Ein CD Grammatiksystem* $\Gamma = (N, T, P_1, \ldots, P_n, S)$ *wird* ***m-linear*** *für ein festes* $m \geq 1$ *genannt, wenn jede Produktion wie folgt gebildet ist:*

$$S \to A_1 \ldots A_{m_0}, \qquad A \to uBv \qquad \textit{oder} \qquad A \to w$$

mit $A, B, A_1, \ldots, A_{m_0} \in (N \setminus \{S\})$, $m_0 \leq m$ und $u, v, w \in T^*$. Wenn ein CD Grammatiksystem m-linear für ein $m \geq 1$ ist, so wird es auch ***metalinear*** *genannt. Wir nennen m die Breite des m-linearen CD Grammatiksystems.*

In diesem Abschnitt werden wir anhand mehrerer Beispiele sehen, daß einige interessante nicht-kontextfreie Sprachen mit metalinearen CD Grammatiksystemen erzeugt werden können. Außerdem wird aufgezeigt, wie man m-linear kontextfreie Sprachen mit metalinearen CD Grammatiksystemen von geringerer Breite erzeugen kann.

Beispiel 6.1 Wir betrachten die Sprache $L_1 = \{a^n b^n a^m b^m a^l b^l \mid l, m, n \geq 1\}$. Das folgende 2-lineare CD Grammatiksystem erzeugt L_1 im t-Modus. L_1 ist allerdings 3-linear, aber nicht 2-linear kontextfrei, wie in [BB93] gezeigt wird. Die Breite der Grammatik kann also unter Umständen reduziert werden, wenn man metalineare CD Grammatiksysteme zur Erzeugung einer Sprache benutzt.

$$\begin{aligned}
\Gamma_1 &= (\{S, A, C, B_1, B_2, B_1', B_2'\}, \{a, b\}, P_1, P_2, P_3, P_4, S),\\
P_1 &= \{S \to B_1 B_2, B_1 \to B_1' a, B_2 \to b B_2'\},\\
P_2 &= \{B_1' \to B_1, B_2' \to B_2\},\\
P_3 &= \{B_1 \to A, B_2 \to C\},\\
P_4 &= \{A \to aAb, C \to aCb, A \to ab, C \to ab\}.
\end{aligned}$$

Die Erzeugung eines Wortes beginnt immer mit der Komponente P_1. Dabei werden die zwei Nichtterminale B_1 und B_2 eingeführt. Danach wird die Anzahl der Symbole a und b in der Mitte des zu erzeugenden Wortes erhöht. Man benutzt dazu die Komponenten P_1 und P_2 im Wechsel und schließt den Vorgang mit P_3 ab. Als nächstes werden mit der Komponente P_4 am Anfang und am Ende des zu erzeugenden Wortes Symbole a und b eingefügt, und die Erzeugung des Wortes wird abgeschlossen. Eine Ableitung sieht also wie folgt aus:

$$\begin{aligned}
S \quad & \overset{t}{\Longrightarrow}_{P_1} B_1' a b B_2' \overset{t}{\Longrightarrow}_{P_2} B_1 a b B_2 \overset{t}{\Longrightarrow}_{P_1} B_1' a a b b B_2'\\
& (\overset{t}{\Longrightarrow}_{P_2} \ldots \overset{t}{\Longrightarrow}_{P_1} \ldots)^{m-2} \overset{t}{\Longrightarrow}_{P_2} B_1 a^m b^m B_2\\
& \overset{t}{\Longrightarrow}_{P_3} A a^m b^m C \overset{t}{\Longrightarrow}_{P_4} a^n b^n a^m b^m a^l b^l.
\end{aligned}$$

Beispiel 6.2 Die Sprache $L_2 = \{ww \mid w \in \{a, b\}^+\}$ ist nicht kontextfrei, kann aber von einem 2-linearen CD Grammatiksystem im $(=2)$-Modus erzeugt werden. Metalineare CD Grammatiksysteme können also nicht nur kontextfreie Sprachen beschreiben.

$$\begin{aligned}
\Gamma_2 &= (\{S, S', A, A', B, B'\}, \{a, b\}, P_1, P_2, P_3, P_4, S),\\
P_1 &= \{S \to AS', S' \to B\},\\
P_2 &= \{A \to bA', B \to bB', A \to b, B \to b\},\\
P_3 &= \{A' \to A, B' \to B\},\\
P_4 &= \{A \to aA', B \to aB', A \to a, B \to a\}.
\end{aligned}$$

Anhand der folgenden Ableitung wird deutlich, wie Γ_2 arbeitet.

$$\begin{aligned} S \quad & \overset{=2}{\Longrightarrow}_{P_1} AB \overset{=2}{\Longrightarrow}_{P_4} aA'aB' \overset{=2}{\Longrightarrow}_{P_3} aAaB \overset{=2}{\Longrightarrow}_{P_2} abA'abB' \\ & \overset{=2}{\Longrightarrow}_{P_3} abAabB \overset{=2}{\Longrightarrow}_{P_4} abaaba. \end{aligned}$$

Zuerst muß die Komponente P_1 benutzt werden. Sie führt die Nichtterminale A und B ein. A wird danach die erste Hälfte des zu erzeugenden Wortes generieren und B die zweite Hälfte. Die obige Ableitung wendet dann die Komponente P_4 an, um jeweils ein a einzufügen. Durch die Restriktionen des $(= 2)$-Modus kann man nur die ersten beiden oder die letzten beiden Produktionen von P_4 zusammen benutzen, da anderenfalls eine Satzform entsteht, die nur ein Nichtterminal enthält und deshalb nicht weiter abgeleitet werden kann. In unserem Fall werden die ersten beiden Produktionen benutzt. Danach muß die Komponente P_3 aktiv werden. Mit P_2 und P_3 wird als nächstes jeweils ein b erzeugt. Dabei kann man beobachten, daß die Komponente P_2 genauso funktioniert wie P_4. Abschließend werden dann noch mit P_4 zwei as eingefügt.

Beispiel 6.3 Die Sprache $L_3 = \{a^n b^n a^n b^n a^n b^n \mid n \geq 0\}$ ist zwar nicht kontextfrei, kann aber von einem 3-linearen CD Grammatiksystem im $(= 2)$-Modus erzeugt werden. Wir werden später sehen, daß L_3 nicht von einem 2-linearen CD Grammatiksystem generiert werden kann.

$$\begin{aligned} \Gamma_3 &= (\{S, A, B, C, S^{(1)}, A^{(1)}, B^{(1)}, C^{(1)}, B^{(2)}, B^{(3)}, B^{(4)}\}, \{a, b\}, \\ &\quad P_1, P_2, P_3, P_4, P_5, P_6, P_7, S) \\ P_1 &= \{S \to S^{(1)}BC, S^{(1)} \to A\}, \\ P_2 &= \{A \to aA^{(1)}b, B \to aB^{(1)}b\}, \\ P_3 &= \{B^{(1)} \to B^{(2)}, C \to aC^{(1)}b\}, \\ P_4 &= \{A^{(1)} \to A, B^{(2)} \to B^{(3)}\}, \\ P_5 &= \{B^{(3)} \to B, C^{(1)} \to C\}, \\ P_6 &= \{A \to \varepsilon, B \to B^{(4)}\}, \\ P_7 &= \{B^{(4)} \to \varepsilon, C \to \varepsilon\}. \end{aligned}$$

Ableitungen von Γ_3 sehen wie folgt aus:

$$\begin{aligned} S \quad & \overset{=2}{\Longrightarrow}_{P_1} ABC \overset{=2}{\Longrightarrow}_{P_2} aA^{(1)}baB^{(1)}bC \overset{=2}{\Longrightarrow}_{P_3} aA^{(1)}baB^{(2)}baC^{(1)}b \\ & \overset{=2}{\Longrightarrow}_{P_4} aAbaB^{(3)}baC^{(1)}b \overset{=2}{\Longrightarrow}_{P_5} aAbaBbaCb \\ & (\overset{=2}{\Longrightarrow}_{P_2} \ldots \overset{=2}{\Longrightarrow}_{P_3} \ldots \overset{=2}{\Longrightarrow}_{P_4} \ldots \overset{=2}{\Longrightarrow}_{P_5})^{n-1} a^n Ab^n a^n Bb^n a^n Cb^n \\ & \overset{=2}{\Longrightarrow}_{P_6} a^n b^n a^n B^{(4)} b^n a^n Cb^n \overset{=2}{\Longrightarrow}_{P_7} a^n b^n a^n b^n a^n b^n. \end{aligned}$$

Am Anfang jeder Ableitung wird mit der Komponente P_1 die Satzform ABC erzeugt. Danach fügen die Komponenten P_2, P_3, P_4 und P_5 vorne, in der Mitte und am Ende des zu erzeugenden Wortes jeweils ein a und ein b ein. Dabei

kann man sehen, daß diese Komponenten durch die Restriktionen des $(= 2)$-Modus nur in der gegebenen Reihenfolge angewendet werden können. Wenn P_2 angewendet wurde, befindet sich in der Satzform ein $A^{(1)}$, ein $B^{(1)}$ und ein C. Da die Komponenten genau zwei Ableitungsschritte machen müssen, kann danach nur Komponente P_3 aktiv werden und so fort. Die Abfolge dieser vier Komponenten kann man beliebig oft wiederholen. Jede Ableitung endet dann mit den Komponenten P_6 und P_7.

6.2 Generative Mächtigkeit

Satz 6.1 *Wenn $f \in \{*, t, =k, \geq k, \leq k \mid k \geq 1\}$, dann ist*

$$\mathcal{L}(CD\text{-}1LIN) = \mathcal{L}(LIN).$$

Beweis: Sei L eine Sprache in $\mathcal{L}(LIN)$ und $G = (N, T, P, S)$ eine linear kontextfreie Grammatik mit $L(G) = L$. Dann gibt es ein CD Grammatiksystem $\Gamma = (N, T, P', S)$ mit nur einer Komponente $P' = P \cup \{A \to A \mid A \in N\}$, die aus den Produktionen der linear kontextfreien Grammatik G und den Kettenproduktionen $A \to A$ besteht. Γ erzeugt L in jedem beliebigen Ableitungsmodus f.

Sei jetzt $\Gamma = (N, T, P_1, \ldots, P_n, S)$ ein lineares CD Grammatiksystem. Wenn $f \in \{*, = 1, \geq 1, \leq k \mid k \geq 1\}$ ist, können wir den Beweis von Satz 4.1 für uneingeschränkte CD Grammatiksysteme in den Modi f und kontextfreie Grammatiken entsprechend modifizieren und erhalten eine linear kontextfreie Grammatik, die $L(\Gamma, f)$ erzeugt.

Sei jetzt $f = t$. Die folgende lineare Grammatik erzeugt $L(\Gamma, t)$.

$$\begin{aligned}
G &= (\{A^{(i)} \mid 1 \leq i \leq n, A \in N\} \cup \{S'\}, T, P', S'), \\
P' &= \{A^{(i)} \to uB^{(i)}v \mid A \to uBv \in P_i,\ A, B \in N,\ u, v \in T^*,\ 1 \leq i \leq n\} \cup \\
&\quad \{A^{(i)} \to v \mid A \to v \in P_i,\ A \in N,\ v \in T^*,\ 1 \leq i \leq n\} \cup \\
&\quad \{A^{(i)} \to A^{(j)} \mid A \notin Dom(P_i),\ A \in Dom(P_j),\ 1 \leq i, j \leq n\} \cup \\
&\quad \{S' \to S^{(i)} \mid S \in Dom(P_i),\ 1 \leq i \leq n\}.
\end{aligned}$$

Durch die Markierungen an den Nichtterminalen wird sichergestellt, daß das einzige Nichtterminal in den Satzformen von G solange durch Produktionen von der gleichen Komponente aus Γ ersetzt wird, bis in dieser Komponente keine Produktion mehr vorhanden ist, die in der aktuellen Satzform angewendet werden kann. Erst dann kann die Markierung des Nichtterminals gewechselt werden. Die Restriktionen des t-Modus werden also korrekt simuliert.

Als nächstes betrachten wir den Fall $f = (=k)$. Die folgende lineare Grammatik erzeugt $L(\Gamma, =k)$.

$$\begin{aligned}
G &= (N, T, P', S), \\
P' &= \{A \to \alpha \mid A \overset{=k}{\Longrightarrow}_{P_i} \alpha,\ 1 \leq i \leq n\}.
\end{aligned}$$

Jede Anwendung einer Produktion in G entspricht hier einer Anwendung von k Produktionen in einer Komponente in Γ.

Es fehlt jetzt noch der Fall $f = (\geq k)$. Jede Ableitung im $(\geq k)$-Modus kann in Teile der Größe k' mit $k \leq k' \leq 2k-1$ unterteilt werden. Die folgende lineare Grammatik erzeugt $L(\Gamma, \geq k)$.

$$\begin{aligned} G &= (N, T, P', S), \\ P' &= \{A \to \alpha \mid A \overset{=k'}{\Longrightarrow}_{P_i} \alpha,\ 1 \leq i \leq n,\ k \leq k' \leq 2k-1\}. \end{aligned}$$

Jede Anwendung einer Produktion in G entspricht dabei einer Anwendung von k' mit $k \leq k' \leq 2k-1$ Produktionen in einer Komponente in Γ. Es wird also gewährleistet, daß mindestens k Produktionen einer Komponente angewendet werden und so die Restriktionen des $(\geq k)$-Modus korrekt simuliert werden. □

Satz 6.2 *Für $m \geq 1$ und den t-Modus gilt*

1. $\mathcal{L}(CD_1\text{-}mLIN, t) = \mathcal{L}(CD_2\text{-}mLIN, t) = \mathcal{L}(mLIN)$,
2. $\mathcal{L}(CD_3\text{-}mLIN, t) = \mathcal{L}(CD\text{-}mLIN, t) = \mathcal{L}(ET0LmLIN)$.

Beweis: Die Konstruktionen aus Satz 4.2 lassen sich auf den m-linearen Fall übertragen. Während der Konstruktionen werden nur lineare Produktionen hinzugefügt und die Breite verändert sich nicht. □

Wir können jetzt zeigen, daß im m-linearen Fall die Klassen der ET0L-Systeme, der Matrixgrammatiken und der CD Grammatiksysteme im Ableitungsmodus $f \in \{t, =k, \geq k \mid k \geq 2\}$ äquivalent sind.

Lemma 6.1 *Wenn $m \geq 1$ und $f \in \{=k, \geq k \mid k \geq 2\}$, dann ist*

$$\mathcal{L}(CD\text{-}mLIN) \subseteq \mathcal{L}(MAT\text{-}mLIN).$$

Beweis: Wenn die Konstruktion für den unbeschränkten Fall aus Satz 4.3. auf metalineare CD Grammatiksysteme angewendet wird, wird die Struktur der Produktionen nicht verändert, und die resultierende Matrixgrammatik ist metalinear und von gleicher Breite. □

Lemma 6.2 *Sei $m \geq 1$. Für jede m-lineare Matrixgrammatik G gibt es eine m-lineare Matrixgrammatik G', die die gleiche Sprache erzeugt und bei der in keiner Satzform ein Nichtterminal mehrfach auftaucht.*

Beweis: Sei $G = (N, T, M, S)$ eine m-lineare Matrixgrammatik. Wir konstruieren jetzt $G' = (N', T, M', S)$. Dabei ist

$$N' = \{S\} \cup \{A^{(i)} \mid A \in N \setminus \{S\},\ 1 \leq i \leq m\}.$$

Bestehe jetzt G aus n Matrizen $m_1, \ldots, m_n$, die jeweils aus k_i Produktionen bestehen, wobei $1 \leq i \leq n$ ist. Es gilt dann $m_i = p_{i,1}, \ldots, p_{i,k_i}$ mit $1 \leq i \leq n$.

Die Menge der Matrizen in der neuen Matrixgrammatik ist

$$M' = \bigcup_{1 \le i \le n} \bigcup_{1 \le j \le k_i} \bigcup_{l_1,\ldots,l_{k_i} \in \{1,\ldots m\}} (f_{l_1}(p_{i,1}), \ldots, f_{l_{k_i}}(p_{i,k_i})).$$

Dabei ist

$$\begin{aligned}
f_{l_j}(p_{i,j}) &= S \to A_1^{(1)} \ldots A_{m_0}^{(m_0)}, \text{ wenn } p_{i,j} = S \to A_1 \ldots A_{m_0},\ m_0 \le m, \\
f_{l_j}(p_{i,j}) &= A^{(l_j)} \to uB^{(l_j)}v, \text{ wenn } p_{i,j} = A \to uBv,\ u, v \in T^*,\ A, B \in N, \\
f_{l_j}(p_{i,j}) &= A^{(l_j)} \to v, \text{ wenn } p_{i,j} = A \to v,\ v \in T^*,\ A \in N.
\end{aligned}$$

Die Produktionen der alten und der neuen Matrixgrammatik unterscheiden sich nur durch die Markierungen an den Nichtterminalen. Es wird also jeweils die gleiche Sprache erzeugt. □

Lemma 6.3 *Sei $m \ge 1$. Für jede m-lineare Matrixgrammatik G gibt es eine äquivalente m-lineare Matrixgrammatik G', bei der jede Matrix genau zwei Produktionen enthält und bei der in keiner Satzform ein Nichtterminal mehrfach auftaucht.*

Beweis: Sei $G = (N, T, M, S)$ eine m-lineare Matrixgrammatik, die dem vorherigen Lemma entspricht und $M = \{m_1, m_2, \ldots, m_n\}$, $m_i = (p_{i,1}, \ldots, p_{i,k_i})$, $1 \le i \le n$. Wir können annehmen, daß eine Matrix, die eine Startproduktion enthält, auch mit dieser anfängt, da nur beim ersten Ersetzungsschritt einer Ableitung das Startsymbol S ersetzt werden kann.

Der Beweis enthält zwei Schritte. Zuerst werden wir eine $(m+1)$-lineare Matrixgrammatik $G'' = (N'', T, M'', S)$ konstruieren, die die obigen Bedingungen erfüllt und die gleiche Sprache wie G erzeugt. Dabei ist $N'' = N \cup \{(i,j),\ 1 \le i \le n,\ 1 \le j \le k_i\}$. Die Menge M'' enthält die folgenden Matrizen:

$$\begin{aligned}
&(S \to A_1 \ldots A_{m_0}(i,1), (i,1) \to (i,2)), \\
&\qquad p_{i,1} = S \to A_1 \ldots A_{m_0},\ k_i \ne 1,\ 1 \le i \le n, \\
&(S \to A_1 \ldots A_{m_0}(i,k_i), (i,k_i) \to (i',1)), \\
&\qquad p_{i,k_i} = S \to A_1 \ldots A_{m_0},\ 1 \le i \le n,\ 1 \le i' \le n \\
&(A \to \alpha, (i,j) \to (i,j+1)), \\
&\qquad A \ne S,\ p_{i,j} = A \to \alpha,\ 1 \le i \le n,\ 1 \le j \le k_i - 1, \\
&(A \to \alpha, (i,k_i) \to (i',1)), \\
&\qquad A \ne S,\ p_{i,k_i} = A \to \alpha,\ 1 \le i \le n,\ 1 \le i' \le n, \\
&(A \to u, (i,k_i) \to \varepsilon), \\
&\qquad A \ne S,\ u \in T^*,\ p_{i,k_i} = A \to u,\ 1 \le i \le n.
\end{aligned}$$

Die neue Matrixgrammatik G'' wurde also um ein Nichtterminal ergänzt, das die Ableitung steuert und anzeigt, welche Produktion aus welcher Matrix gerade angewendet wird. Man kann sehen, daß jeder Ableitung in G eine Ableitung in G'' entspricht, die das gleiche Wort erzeugt. Das Nichtterminal (i,j)

stellt in G'' sicher, daß Produktionen aus verschiedenen Matrizen nicht vermischt werden können und auch in der gegebenen Reihenfolge ausgeführt werden müssen. Fehlt das Nichtterminal (i, j), so ist die Ableitung entweder beendet, oder die Ableitung ist blockiert, weil keine Matrix mehr angewendet werden kann. Es gibt also auch für jede Ableitung in G'' eine entsprechende Ableitung in G, die das gleiche Wort erzeugt. Damit ist $L(G) = L(G'')$.

Als nächstes konstruieren wir aus der Matrixgrammatik G'' eine Matrixgrammatik $G' = (N', T, M', S)$, die die geforderten Eigenschaften hat und m-linear ist. Dafür wird das Nichtterminal (i, j) mit einem anderen Nichtterminal zusammengefaßt. Es ist jetzt

$$N' = N \cup \{[A, (i, j)] \mid A \in N,\ 1 \le i \le n,\ 1 \le j \le k_i\}.$$

Die Menge M' besteht aus den Matrizen

$$\begin{aligned}
&(S \to A_1 \dots [A_{m_0}, (i, 1)], [A_{m_0}, (i, 1)] \to [A_{m_0}, (i, 2)]),\\
&\qquad (S \to A_1 \dots A_{m_0}(i, 1),\ (i, 1) \to (i, 2)) \in M'',\\
&(S \to A_1 \dots [A_{m_0}, (i, k_i)], [A_{m_0}, (i, k_i)] \to [A_{m_0}, (i', 1)]),\\
&\qquad (S \to A_1 \dots A_{m_0}(i, k_i),\ (i, k_i) \to (i', 1)) \in M'',\\
&([A, (i, j)] \to A, B \to [B, (i, j)]),\\
&\qquad A, B \in N,\ 1 \le i \le n,\ 1 \le j \le k_i,\\
&([A, (i, j)] \to u[B, (i, j)]v, [B, (i, j)] \to [B, (i, j + 1)]),\\
&\qquad (A \to uBv, (i, j) \to (i, j + 1)) \in M'',\ A, B \in N,\ u, v \in T^*,\\
&([A, (i, k_i)] \to u[B, (i, k_i)]v, [B, (i, k_i)] \to [B, (i', 1)]),\\
&\qquad (A \to uBv, (i, k_i) \to (i', 1)) \in M'',\ A, B \in N,\ u, v \in T^*,\\
&([A, (i, j)] \to u, B \to [B, (i, j + 1)]),\\
&\qquad (A \to u, (i, j) \to (i, j + 1)) \in M'',\ A, B \in N,\ u, v \in T^*,\\
&([A, (i, k_i)] \to u, B \to [B, (i', 1)]),\\
&\qquad (A \to u, (i, k_i) \to (i', 1)) \in M'',\ A, B \in N,\ u, v \in T^*,\\
&([A, (i, k_i)] \to [A, (i, k_i)], [A, (i, k_i)] \to u),\\
&\qquad (A \to u, (i, k_i) \to \varepsilon) \in M''.
\end{aligned}$$

In der Matrixgrammatik G' ist das Nichtterminal (i, j) nicht wie in G'' am rechten Rand einer Satzform zu finden, sondern mit einem anderen Nichtterminal zusammengefaßt. Die Matrizen von G' sind so konstruiert, daß eine Matrix nur mit der Ersetzung eines Nichtterminals beginnen kann, wenn dieses Nichtterminal mit dem Nichtterminal (i, j) zusammengefaßt ist. Um jetzt alle Matrizen anwenden zu können, muß man den Zusatz (i, j) von einem zum anderen Nichtterminal verschieben können. Das passiert mit der dritten Matrix aus G'. Ansonsten arbeitet G' wie G'' und erzeugt die gleiche Sprache. Allerdings ist G' nur noch m-linear und erfüllt deshalb die Bedingungen des Lemmas. □

Lemma 6.4 *Sei $m \ge 1$. Für jede m-lineare Matrixgrammatik G gibt es eine äquivalente m-lineare Matrixgrammatik G', bei der jede Matrix genau zwei*

Produktionen enthält und bei der in keiner Satzform ein Nichtterminal mehrfach auftaucht. Außerdem gilt für jede Produktion $A \to \alpha$, daß α das Nichtterminal A nicht enthält. Weiterhin ist $L(G) = L(G')$.

Beweis: Sei $G = (N, T, M, S)$ eine m-lineare Matrixgrammatik, deren Form Lemma 6.3 entspricht. Wir konstruieren jetzt eine weitere Matrixgrammatik $G' = (N', T, M', S)$, die die gleiche Sprache erzeugt und die obige Form hat. Dabei ist $N' = N \cup N^{(1)} \cup N^{(2)}$. Die Homomorphismen (1) und (2) sind definiert wie im 2. Kapitel. Außerdem enthält M' die folgenden Matrizen.

$$
\begin{array}{l}
(A^{(1)} \to A^{(2)}, A^{(2)} \to A),\ A \in N, \\
(S \to A_1 \dots A_{m_0}, A \to \alpha^{(1)}),\ (S \to A_1 \dots A_{m_0}, A \to \alpha) \in M, \\
(A \to \alpha^{(1)}, B \to \beta^{(1)}),\ (A \to \alpha, B \to \beta) \in M,\ A, B \neq S, \\
(A \to \alpha^{(1)}, B^{(1)} \to \beta^{(1)}),\ (A \to \alpha, B \to \beta) \in M,\ A, B \neq S.
\end{array}
$$

Die Ableitungen in M' verlaufen genauso wie in M. Wenn nötig werden die Markierungen (1) mit dem ersten Typ von Matrizen aus M' entfernt. Man kann also sehen, daß $L(G) = L(G')$ ist. □

Lemma 6.5 *Sei $m \geq 1$. Für jedes m-lineare ET0L-System G gibt es ein m-lineares ET0L-System G', das die gleiche Sprache erzeugt und bei dem in keiner Satzform ein Nichtterminal mehrfach auftaucht.*

Beweis: Sei $G = (V, T, P_1, \dots, P_n, S)$ ein m-lineares ET0L-System. Wir konstruieren jetzt $G' = (V', T, P'_1, \dots, P'_n, S)$. Dabei können wir annehmen, daß die Menge der aktiven Symbole und die Menge der Nichtterminale $N = V \setminus T$ übereinstimmt. Es ist $V' = N' \cup T$ und

$$N' = \{S\} \cup \{A^{(i)} \mid A \in N \setminus \{S\},\ 1 \leq i \leq m\}.$$

Die Menge der Tabellen im neuen ET0L-System ist für $1 \leq i \leq n$

$$
\begin{array}{rcl}
P'_i & = & \{S \to A_1^{(1)} \dots A_{m_0}^{(m_0)} \mid S \to A_1 \dots A_{m_0} \in P_i,\ m_0 \leq m\} \cup \\
& & \{B^{(j)} \to uC^{(j)}v \mid B \to uCv \in P_i,\ B, C \in N,\ u, v \in T^*, 1 \leq j \leq m\} \cup \\
& & \{B^{(j)} \to u \mid B \to u \in P_i,\ B \in N,\ u \in T^*, 1 \leq j \leq m\}.
\end{array}
$$

Die Tabellen des alten und des neuen ET0L-Systems unterscheiden sich nur durch die Markierungen an den Nichtterminalen. Es wird also jeweils die gleiche Sprache erzeugt. □

Lemma 6.6 *Sei $m \geq 1$. Für jedes m-lineare ET0L-System G gibt es eine m-lineare Matrixgrammatik G', die die gleiche Sprache erzeugt.*

Beweis: Sei $G = (V, T, P_1, \dots, P_n, S)$ ein m-lineares ET0L-System. Nach dem vorherigen Lemma können wir annehmen, daß in keiner Satzform ein Nichtterminal mehrfach auftaucht. Sei jetzt

$$
\begin{array}{rcl}
N' & = & \{[A_1 \dots A_{m_0}, z_j] \mid A_1, \dots, A_{m_0} \in N, \\
& & \quad m_0 \leq m,\ 1 \leq j \leq m_0,\ 1 \leq z_1 < z_2 < \dots < z_m \leq m\}.
\end{array}
$$

Wir konstruieren jetzt eine Menge M' von Matrizen für die m-lineare Matrixgrammatik.

Wenn für ein i mit $1 \leq i \leq n$ und $A_1, \ldots, A_{m_0} \in N \setminus S$ gilt, daß

$$A_1 \ldots A_{m_0} \Rightarrow_{P_i} u_1 B_1 v_1 u_2 B_2 v_2 \ldots u_{m_0} B_{m_0} v_{m_0}$$

ist, wobei $A_j \Rightarrow_{P_i} u_j B_j v_j$, $u_j, v_j \in T^*$, $B_j \in N^{\leq 1}$ und $1 \leq j \leq m_0$ gilt, dann fügen wir die Matrix

$$\begin{aligned} m \quad : \quad & ([A_1 \ldots A_{m_0}, 1] \rightarrow u_1 [B_1 \ldots B_{m_0}, 1] v_1, \\ & [A_1 \ldots A_{m_0}, 2] \rightarrow u_2 [B_1 \ldots B_{m_0}, 2] v_2, \ldots, \\ & [A_1 \ldots A_{m_0}, m_0] \rightarrow u_{m_0} [B_1 \ldots B_{m_0}, m_0] v_{m_0}) \end{aligned}$$

zu M' hinzu. Dabei ist $[B_1 \ldots B_{m_0}, j] = \varepsilon$, wenn $B_j = \varepsilon$ ist.

Wenn für ein i mit $1 \leq i \leq n$ gilt, daß

$$S \Rightarrow_{P_i} A_1 \ldots A_{m_0}$$

mit $m_0 \leq m$, dann fügen wir die Matrix

$$m : ([S, 1] \Rightarrow [A_1 \ldots A_{m_0}, 1] \ldots [A_1 \ldots A_{m_0}, m_0])$$

zu M' hinzu.

Die Matrixgrammatik $G' = (N', T, M', [S, 1])$ erzeugt $L(G)$, da mit jeder Matrix ein paralleler Ersetzungsschritt von G simuliert wird. Durch die Form der Nichtterminale wird sichergestellt, daß eine Matrix nur angewendet werden kann, wenn sie alle in der Satzform vorhandenen Nichtterminale ersetzen kann. Da die Produktionen in ihrer Struktur nicht verändert wurden, ist G' auch m-linear. □

Satz 6.3 *Für $m \geq 1$ und $d \in \{=k, \geq k \mid k \geq 2\}$ gilt*

$$\mathcal{L}(\textit{MAT-mLIN}) = \mathcal{L}(\textit{CD-mLIN}, d) = \mathcal{L}(\textit{CD-mLIN}, t) = \mathcal{L}(\textit{ET0L-mLIN}).$$

Beweis: Aus den vorangegangenen Lemmata ergibt sich, daß wir aus jedem m-linearen CD Grammatiksystem Γ, das im Modus $d \in \{=k, \geq k \mid k \geq 2\}$ arbeitet, eine m-lineare Matrixgrammatik G konstruieren können, die die gleiche Sprache erzeugt. Aus dieser können wir dann eine äquivalente Matrixgrammatik $G' = (N, T, P, S)$ erzeugen, in deren Satzformen kein Nichtterminal mehrfach vorkommt und deren Matrizen jeweils zwei Produktionen enthalten. Außerdem enthält nach Lemma 6.4 bei jeder Produktion $A \rightarrow \alpha$ der String α das Nichtterminal A nicht.

Wir können jetzt ein CD Grammatiksystem $\Gamma' = (N, T, P_1, \ldots, P_n, S)$ konstruieren, für das $L(G') = L(\Gamma', = 2)$ gilt. Dazu wandeln wir die Matrizen $m_1, \ldots, m_n$ aus M unverändert in Komponenten um wie folgt:

$$P_i = \{A \rightarrow \alpha, B \rightarrow \beta\},\ m_i : (A \rightarrow \alpha, B \rightarrow \beta),\ 1 \leq i \leq n.$$

Durch die besondere Form von G' ist gewährleistet, daß jede Produktion nur einmal angewendet werden kann. Also erzeugt Γ' im $(= 2)$-Modus die gleiche Sprache. Da beim CD Grammatiksystem Γ pro Komponente genau zwei Produktionen angewendet werden können, gilt auch $L(\Gamma, = 2) = L(\Gamma, t)$.

Mit der gleichen Technik wie im Beweis von Satz 4.4 kann Γ' in ein CD Grammatiksystem umgewandelt werden, das im $(=k)$- oder $(\geq k)$-Modus für $k \geq 2$ arbeitet und die gleiche Sprache erzeugt.

Aus einem m-linearen CD Grammatiksystem im t-Modus können wir ein m-lineares ET0L-System konstruieren, das die gleiche Sprache erzeugt. Außerdem kann man aus einem m-linearen ET0L-System ein m-lineares CD Grammatiksystem erzeugen, das im t-Modus arbeitet und die gleiche Sprache erzeugt. Beide Konstruktionen sind für den unbeschränkten Fall im Beweis von Satz 4.2 zu finden und verändern nicht die Breite der CD Grammatiksysteme beziehungsweise ET0L-Systeme. Sie können deshalb auch im m-linearen Fall verwendet werden.

Nach Lemma 6.6 können wir für jedes m-lineare ET0L-System eine m-lineare Matrixgrammatik konstruieren, die die gleiche Sprache erzeugt. Insgesamt erhalten wir somit den Satz. □

Wenn die generative Mächtigkeit eines Modells untersucht werden soll, benötigt man die Möglichkeit festzustellen, ob eine bestimmte Sprache in einer bestimmten Sprachklasse enthalten ist oder nicht. Wir werden in diesem Abschnitt sehen, daß es für metalineare CD Grammatiksysteme zu diesem Zweck mehrere Pumpinglemmata gibt.

Lemma 6.7 *Sei $m \geq 1$, T ein endliches Alphabet, $f \in \{t, = k, \geq k \mid k \geq 2\}$ und $L \subseteq T^*$ mit $L \in \mathcal{L}(CD\text{-}mLIN, f)$ eine unendliche Sprache. Dann gibt es ein $h \in \mathbb{N}$, so daß für alle $w \in L$ mit $|w| \geq h$ eine Zerlegung $w = c_1 \dots c_m$ existiert und folgendes gilt:*

1. $c_i = x_i u_i y_i v_i z_i$, $1 \leq i \leq m$,
2. $u_1 v_1 u_2 v_2 \dots u_m v_m \neq \varepsilon$,
3. *für alle i_0 mit $1 \leq i_0 \leq m$ und $|c_{i_0}| \geq h/m$ ist*

   $$u_{i_0} v_{i_0} \neq \varepsilon \text{ und } |x_{i_0} u_{i_0} v_{i_0} z_{i_0}| \leq h/m,$$

4. *für alle j mit $j \geq 0$ ist*

   $$x_1 u_1^j y_1 v_1^j z_1 x_2 u_2^j y_2 v_2^j z_2 \dots x_m u_m^j y_m v_m^j z_m \in L.$$

Beweis: Zuerst werden wir das Lemma für den t-Modus beweisen. Sei L eine unendliche Sprache in $\mathcal{L}(CD\text{-}mLIN, t)$ und $\Gamma = (N, T, P_1, \dots, P_n, S)$ ein CD Grammatiksystem in $(CD\text{-}mLIN, t)$ mit $L(\Gamma, t) = L$. Weiterhin sei s die maximale Anzahl von Terminalen in Produktionen aus Γ, p die maximale Anzahl von Produktionen mit Terminalen in einer Komponente und r die Anzahl der Nichtterminale in Γ. Wenn jetzt ein Wort $w \in L$ aus $|w|$ Terminalen

besteht, dann wurden höchstens $|w|/s$ Produktionen mit Terminalen in einer Ableitung von w und dem entsprechenden Ableitungsbaum benutzt. Sei jetzt $h = (p+1)((r+1)^m + 1) \cdot s \cdot m$ und $w \in L$ mit $|w| \geq h$. Sei D eine Ableitung von w in Γ und t der entsprechende Ableitungsbaum. Das Wort w ist dann in m Teile $w = c_1 \dots c_m$ geteilt, die sich durch die erste Produktion der Ableitung $S \Rightarrow A_1 \dots A_m$ und die linearen Teilbäume t_i des Ableitungsbaums t mit Wurzel A_i und Blattfolge c_i, $1 \leq i \leq m$, ergeben. Außerdem gibt es mindestens ein $c_{i_0} \in \{c_1, \dots, c_m\}$ mit $|c_{i_0}| \geq (p+1)((r+1)^m + 1) \cdot s = h/m$. Um jetzt den ersten Teil des Lemmas zu zeigen, betrachten wir den Teilbaum t_{i_0} mit der Wurzel A_{i_0}, der c_{i_0} erzeugt. Nur lineare Produktionen werden in diesem Teilbaum angewendet. Wenn wir jetzt die ersten $(p+1)((r+1)^m+1)$ Produktionen mit Terminalen untersuchen, können wir folgendes feststellen:

1. Fall: Bei der Ableitung der ersten $(p+1)((r+1)^m+1)$ Produktionen mit Terminalen von t_{i_0} hat eine Komponente mehr als p Produktionen angewendet. Es gibt dann zwei Knoten vt_1 und vt_2, die die folgenden Eigenschaften haben:

- Beide Knoten sind mit dem gleichen Nichtterminal bezeichnet, und die gleichen Produktionen wurden an ihnen angewendet.
- Der Knoten vt_1 befindet sich näher an der Wurzel als der Knoten vt_2.
- Der Pfad von vt_1 nach vt_2 enthält höchstens $p+1$ Produktionen mit Terminalen.

In diesem Fall können wir innerhalb einer Komponente die Produktionen auf dem Pfad des betrachteten Teilbaums mehrfach anwenden. Für $1 \leq g \leq m$, $g \neq i_0$ sind die Strings $x_g, u_g, v_g, z_g = \varepsilon$, $c_{i_0} = x_{i_0} u_{i_0} y_{i_0} v_{i_0} z_{i_0}$ und

$$\begin{aligned} w &= y_1 \dots x_{i_0} u_{i_0} y_{i_0} v_{i_0} z_{i_0} \dots y_m \\ &= x_1 u_1 y_1 v_1 z_1 \dots x_{i_0} u_{i_0} y_{i_0} v_{i_0} z_{i_0} \dots x_m u_m y_m v_m z_m. \end{aligned}$$

Wenn die Produktionen auf dem Pfad von vt_1 zu vt_2 dann j mal hintereinander angewendet werden, erhalten wir

$$\begin{aligned} w' &= y_1 \dots x_{i_0} u_{i_0}^j y_{i_0} v_{i_0}^j z_{i_0} \dots y_m \\ &= x_1 u_1^j y_1 v_1^j z_1 \dots x_{i_0} u_{i_0}^j y_{i_0} v_{i_0}^j z_{i_0} \dots x_m u_m^j y_m v_m^j z_m \end{aligned}$$

mit $w' \in L$. Außerdem ist $u_{i_0} v_{i_0} \neq \varepsilon$ und $|x_{i_0} u_{i_0} v_{i_0} z_{i_0}| \leq (p+1)s \leq h/m$.

2. Fall: Bei der Ableitung der ersten $(p+1)((r+1)^m+1)$ Produktionen mit Terminalen von t_{i_0} hat keine Komponente mehr als p Produktionen angewendet. Dann muß es aber mehr als $(r+1)^m + 1$ Wechsel von Komponenten während der ersten $(p+1)((r+1)^m+1)$ Produktionen mit Terminalen im Teilbaum mit Wurzel A_{i_0} und Blattfolge c_{i_0} gegeben haben.

Wenn wir jetzt die höchstens m linearen Unterbäume eines unvollständigen Ableitungsbaums für das Wort w in dem Moment betrachten, in dem ein Komponentenwechsel stattfindet, dann befindet sich an den Blättern jedes Teilbaums höchstens ein Nichtterminal. Im folgenden wird solch ein unvollständiger Ableitungsbaum durch eine Sequenz mit m Positionen dargestellt. Jede

Position ist mit dem Symbol $\times$ markiert, wenn der entsprechende Unterbaum kein mit einem Nichtterminal markiertes Blatt besitzt. Besitzt der entsprechende Unterbaum ein mit einem Nichtterminal markiertes Blatt, dann ist die entsprechende Stelle mit diesem Nichtterminal markiert. Es gibt nur $(r+1)^m$ verschiedene Sequenzen. Eine dieser Sequenzen muß also während eines Komponentenwechsels in der Ableitung der ersten $(p+1)((r+1)^m+1)$ Produktionen von t_{i_0} mit Terminalen doppelt erscheinen. Seien

$$K_1 = (\times_1, \ldots, \times_{i_1-1}, B_{i_1}, \times_{i_1+1}, \ldots, \times_{i_p-1}, B_{i_p}, \times_{i_p+1}, \ldots, \times_m)$$

und

$$K_2 = (\times_1, \ldots, \times_{i_1-1}, B_{i_1}, \times_{i_1+1}, \ldots, \times_{i_p-1}, B_{i_p}, \times_{i_p+1}, \ldots, \times_m)$$

diese beiden Sequenzen, wobei $B_{i_1}, \ldots, B_{i_p} \in N$ und jede Position wie oben beschrieben für einen linearen Unterbaum des Ableitungsbaums von w steht. In diesem Fall ist $x_g, u_g, v_g, z_g = \varepsilon,\ 1 \leq g \leq m,\ g \notin \{i_1, \ldots, i_p\}$, und wir haben

$$\begin{aligned} w &= y_1 \ldots y_{i_1-1} x_{i_1} u_{i_1} y_{i_1} v_{i_1} z_{i_1} y_{i_1+1} \ldots y_{i_p-1} x_{i_p} u_{i_p} y_{i_p} v_{i_p} z_{i_p} y_{i_p+1} \ldots y_m \\ &= x_1 u_1 y_1 v_1 z_1 \ldots \\ &\quad x_{i_1-1} u_{i_1-1} y_{i_1-1} v_{i_1-1} z_{i_1-1} x_{i_1} u_{i_1} y_{i_1} v_{i_1} z_{i_1} x_{i_1+1} u_{i_1+1} y_{i_1+1} v_{i_1+1} z_{i_1+1} \ldots \\ &\quad x_{i_p-1} u_{i_p-1} y_{i_p-1} v_{i_p-1} z_{i_p-1} x_{i_p} u_{i_p} y_{i_p} v_{i_p} z_{i_p} x_{i_p+1} u_{i_p+1} y_{i_p+1} v_{i_p+1} z_{i_p+1} \ldots \\ &\quad x_m u_m y_m v_m z_m. \end{aligned}$$

Wenn die Produktionen, die den Baum von Konfiguration

$$K_1 = (\times_1, \ldots, \times_{i_1-1}, B_{i_1}, \times_{i_1+1}, \ldots, \times_{i_p-1}, B_{i_p}, \times_{i_p+1}, \ldots, \times_m)$$

zur Konfiguration

$$K_2 = (\times_1, \ldots, \times_{i_1-1}, B_{i_1}, \times_{i_1+1}, \ldots, \times_{i_p-1}, B_{i_p}, \times_{i_p+1}, \ldots, \times_m)$$

überführen, j mal angewendet werden, erhält man das folgende Wort

$$\begin{aligned} w' &= y_1 \ldots y_{i_1-1} x_{i_1} u_{i_1}^j y_{i_1} v_{i_1}^j z_{i_1} y_{i_1+1} \ldots y_{i_p-1} x_{i_p} u_{i_p}^j y_{i_p} v_{i_p}^j z_{i_p} y_{i_p+1} \ldots y_m \\ &= x_1 u_1^j y_1 v_1^j z_1 \ldots \\ &\quad x_{i_1-1} u_{i_1-1}^j y_{i_1-1} v_{i_1-1}^j z_{i_1-1} x_{i_1} u_{i_1}^j y_{i_1} v_{i_1}^j z_{i_1} x_{i_1+1} u_{i_1+1}^j y_{i_1+1} v_{i_1+1}^j z_{i_1+1} \ldots \\ &\quad x_{i_p-1} u_{i_p-1}^j y_{i_p-1} v_{i_p-1}^j z_{i_p-1} x_{i_p} u_{i_p}^j y_{i_p} v_{i_p}^j z_{i_p} x_{i_p+1} u_{i_p+1}^j y_{i_p+1} v_{i_p+1}^j z_{i_p+1} \ldots \\ &\quad x_m u_m^j y_m v_m^j z_m \end{aligned}$$

mit $w' \in L$. Außerdem gilt für $i_0 \in \{i_1, \ldots, i_p\}$, daß $x_{i_0} u_{i_0}$ und $v_{i_0} z_{i_0}$ Teilstrings von c_{i_0} mit den Eigenschaften $u_{i_0} v_{i_0} \neq \varepsilon$ und $|x_{i_0} u_{i_0} v_{i_0} z_{i_0}| \leq p((r+1)^m+1)s \leq h/m$ sind.

Wenn jetzt $d \in \{=k, \geq k \mid k \geq 2\}$ und L eine unendliche Sprache in der Klasse $\mathcal{L}(\textit{CD-mLIN}, d)$ ist, dann gibt es nach Satz 6.3 auch ein CD Grammatiksystem in $(\textit{CD-mLIN}, t)$, das L erzeugt. Das Pumpinglemma gilt demnach auch für alle anderen Ableitungsmodi. □

Lemma 6.8 *Sei $m \geq 1$, T ein endliches Alphabet, $f \in \{t, =k, \geq k \mid k \geq 2\}$ und $L \subseteq T^*$ mit $L \in \mathcal{L}$(CD-mLIN, f) eine unendliche Sprache. Dann gibt es ein $h \in \mathbb{N}$, so daß für alle w mit $|w| \geq h$ eine Zerlegung $w = c_1 \dots c_m$ existiert. Für alle i_0, $1 \leq i_0 \leq m$, mit $|c_{i_0}| \geq h/m$ und jedes Teilwort c'_{i_0} von c_{i_0} mit $|c'_{i_0}| \geq h/m$ gibt es ein $\gamma \in T^*$, und es gilt*

1. *$c'_{i_0}\gamma$ ist ein Teilwort von c_{i_0},*
2. *$c'_{i_0}\gamma = xuyvz$, $uv \neq \varepsilon$ und $xuvz$ enthält höchstens h/m Symbole von c'_{i_0},*
3. *xu ist ein Teilwort der ersten h/m Terminale von $c'_{i_0}\gamma$, vz ist ein Teilwort der letzten h/m Terminale von $c'_{i_0}\gamma$ und, wenn u leer ist, dann ist v ein Teilwort der letzten h/m Terminale von c'_{i_0},*
4. *für alle j mit $j \geq 0$ gibt es $\phi, \phi' \in T^*$ mit $\phi x u^j y v^j z \phi' \in L$,*

oder

1. *$\gamma c'_{i_0}$ ist ein Teilwort von c_{i_0},*
2. *$\gamma c'_{i_0} = xuyvz$, $uv \neq \varepsilon$ und $xuvz$ enthält höchstens h/m Symbole von c'_{i_0},*
3. *xu ist ein Teilwort der ersten h/m Terminale von $\gamma c'_{i_0}$, vz ist ein Teilwort der ersten h/m Terminale von $\gamma c'_{i_0}$ und, wenn v leer ist, dann ist u ein Teilwort der letzten h/m Terminale von c'_{i_0},*
4. *für alle j mit $j \geq 0$ gibt es $\phi, \phi' \in T^*$ mit $\phi x u^j y v^j z \phi' \in L$.*

Beweis: Zuerst werden wir das Lemma für den t-Modus beweisen. Sei L eine unendliche Sprache in $\mathcal{L}$(CD-mLIN, t) und $\Gamma = (N, T, P_1, \dots, P_n, S)$ ein CD Grammatiksystem in (CD-mLIN, t) mit $L(\Gamma, t) = L$. Weiterhin sei s die maximale Anzahl von Terminalen in Produktionen aus Γ, p die maximale Anzahl von Produktionen mit Terminalen in einer Komponente und r die Anzahl der Nichtterminale in Γ. Sei jetzt die Konstante $h = 2 \cdot (p+1)((r+1)^m + 1) \cdot s \cdot m$. Wie im Beweis zum Lemma 6.7 gezeigt, ergibt sich c_{i_0} aus einem linearen Teilbaum t_{i_0} des Ableitungsbaums t von w. Jedes Teilwort c'_{i_0} von c_{i_0} können wir mit zwei Strings γ, $\gamma' \in T^*$, $|\gamma'| < s$ zu $\gamma' c_{i_0} \gamma$ oder $\gamma c_{i_0} \gamma'$ erweitern, so daß wir ein Wort erhalten, das durch einen Unterbaum t'_{i_0} von t_{i_0} erzeugt wird. Da $|c'_{i_0}| > h/m$, besteht t'_{i_0} aus mindestens $h/m/s$ Produktionen mit Terminalen, die zur Erzeugung von c'_{i_0} beitragen. Wir werden nur den Fall $yield(t'_{i_0}) = \gamma' c'_{i_0} \gamma$ behandeln, da der andere Fall analog bewiesen werden kann.

Im folgenden wird der kürzeste Pfad untersucht, der mit der zweiten Produktion beginnt und $2 \cdot ((r+1)^m + 1) \cdot (p+1)$ Produktionen, die zur Erzeugung von c'_{i_0} beitragen, enthält. Dieser Pfad kann auch Produktionen enthalten, die nur Symbole von γ erzeugen. Höchstens $2 \cdot ((r+1)^m + 1) \cdot (p+1) \cdot s$ Symbole von c'_{i_0} werden auf diesem Pfad generiert. Außerdem können wir einen Unterpfad finden, der entweder $((r+1)^m + 1) \cdot (p+1)$ Produktionen enthält, die zu den ersten $((r+1)^m + 1) \cdot (p+1) \cdot s$ Symbolen von c'_{i_0} beitragen, und mit einer Produktion anfängt, die zu den ersten $((r+1)^m + 1) \cdot (p+1) \cdot s$ Symbolen von c'_{i_0} beiträgt, oder $((r+1)^m + 1) \cdot (p+1)$ Produktionen enthält, die zu den letzten

$((r+1)^m+1)\cdot(p+1)\cdot s$ Symbolen von c'_{i_0} beitragen, und mit einer Produktion anfängt, die zu den letzten $((r+1)^m+1)\cdot(p+1)\cdot s$ Symbolen von c'_{i_0} beiträgt.

Im Beweis von Lemma 6.7 haben wir gezeigt, daß in einem Unterbaum aus $((r+1)^m+1)\cdot(p+1)$ Produktionen entweder eine Komponente mehr als p Produktionen anwenden muß (1. Fall), oder mehr als $(r+1)^m+1$ Komponentenwechsel stattfinden müssen (2. Fall). In beiden Fällen gibt es zwei Knoten vt_1 und vt_2, die mit dem gleichen Nichtterminal bezeichnet sind. Man kann analog zeigen, daß in dem oben betrachteten Unterpfad entweder eine Komponente mehr als p Produktionen, die zu dem ersten oder letzten Terminalen von c'_{i_0} beitragen, anwenden muß, oder mindestens $(r+1)^m+1$ Komponentenwechsel stattfinden. In beiden Fällen gibt es zwei Knoten vt_1 und vt_2, die mit dem gleichen Nichtterminal bezeichnet sind. Mindestens eine Produktion auf dem Pfad von vt_1 nach vt_2 trägt zur Generierung von c'_{i_0} bei.

Es ist also $\gamma' c'_{i_0}\gamma = x'uyvz$. Da wir die erste Produktion von t'_{i_0} im folgenden nicht benutzen und da γ' so gewählt werden kann, daß es von der ersten Produktion in t'_{i_0} erzeugt wird, können wir im weiteren auch den Teilstring $c'_{i_0}\gamma = xuyvz$ betrachten, wobei $x' = \gamma' x$ ist.

Die Teilstrings xu und vz werden von höchstens $2\cdot((r+1)^m+1)\cdot(p+1)$ Produktionen, die zu c'_{i_0} beitragen, und eventuell einigen anderen Produktionen, die nur zu γ beitragen, erzeugt. Auf dem Pfad von vt_1 nach vt_2 befindet sich aber mindestens eine Produktion, die zu c'_{i_0} beiträgt. Also enthält uv mindestens ein Symbol aus c'_{i_0} und $xuvz$ enthält höchstens $2\cdot((r+1)^m+1)\cdot(p+1)\cdot s = h/m$ Symbole aus c'_{i_0}. Die anderen Symbole von c'_{i_0} sind in y enthalten.

Wenn u leer ist, dann beginnt der oben betrachtete Unterpfad mit einer Produktion, die zu den letzten $((r+1)^m+1)\cdot(p+1)\cdot s$ Symbolen von c'_{i_0} beiträgt, und enthält $((r+1)^m+1)\cdot(p+1)$ Produktionen, die zum letzten Teil von c'_{i_0} beitragen. In diesem Fall ist v ein Teilstring der letzten $2\cdot((r+1)^m+1)\cdot(p+1)\cdot s \leq h/m$ Symbole von c'_{i_0}. Da wir den Pfad von der zweiten bis zur $2\cdot((r+1)^m+1)\cdot(p+1)$ten Produktion, die zu c'_{i_0} beitragen, analysieren, ist xu ein Teilstring der ersten h/m Terminale von $c'_{i_0}\gamma$ und vz ein Teilstring der letzten h/m Terminale von $c'_{i_0}\gamma$.

Wenn die Produktionen auf dem Pfad von vt_1 nach vt_2 j-mal angewendet werden, erhalten wir xu^jyv^jz. Im ersten Teil des Beweises haben wir gezeigt, daß es möglich ist, die Produktionen auf dem Pfad von vt_1 nach vt_2 j-mal anzuwenden und ein Wort in L zu erhalten. Es gibt also für alle j mit $j \geq 0$ Zeichenketten $\phi, \phi' \in T^*$, so daß $\phi xu^jyv^jz\phi' \in L$ ist.

Wenn jetzt $d \in \{=k, \geq k \mid k \geq 2\}$ und L eine unendliche Sprache in der Klasse $\mathcal{L}(\textit{CD-mLIN}, d)$ ist, dann gibt es nach Satz 6.3 auch ein CD Grammatiksystem in $(\textit{CD-mLIN}, t)$, das L erzeugt. Das Pumpinglemma gilt demnach auch für alle anderen Ableitungsmodi. □

Mit Hilfe der eben gezeigten Pumpinglemmata können wir nun einige Eigenschaften von metalinearen CD Grammatiksystemen beweisen.

Satz 6.4 *Sei $f \in \{t, =k, \geq k \mid k \geq 2\}$. Die kontextfreie Sprache*

$$L = \{a^n b^n \mid n \geq 1\}^*$$

ist nicht in der Klasse $\mathcal{L}$(CD-METALIN, f) enthalten.

Beweis: Wir beweisen den Satz mit Hilfe von Lemma 6.8. Angenommen, L wäre in $\mathcal{L}$(CD-METALIN, f), dann müßte L auch in $\mathcal{L}$(CD-mLIN, f) für ein $m \geq 1$ sein. Sei h die Konstante aus Lemma 6.8. Wir betrachten das Wort $w = (a^h b^h)^{3m}$. Es zerfällt nun in m Teilwörter, von denen mindestens eines größer als $6h$ ist. Dieses Teilwort ist c_{i_0} aus Lemma 6.8. Der String c_{i_0} enthält einen Teilstring $c'_{i_0} = a^h b^h a^h b^h$. Im ersten Fall des Lemmas läßt sich sich c'_{i_0} mit einem γ ergänzen und $c'_{i_0}\gamma$ ist auch ein Teilstring von c_{i_0}. Es gibt eine Aufteilung $xuyvz$ des Wortes $c'_{i_0}\gamma$. Dabei ist entweder xu ein Teilwort der ersten h/m Zeichen, oder u ist leer und vz ein Teilwort der letzten h/m Zeichen von c'_{i_0}. Außerdem ist $|uv| > 0$ und $xuvz$ enthält höchstens h/w Symbole von c'_{i_0}. Da y die übrigen Symbole von c'_{i_0} enthalten muß, kann entweder in der ersten Gruppe von Symbolen a gepumpt werden, ohne daß die zugehörige Gruppe von Symbolen b entsprechend vergrößert wird oder in der letzten Gruppe von Symbolen b gepumpt werden, ohne daß die zugehörige Gruppe von Symbolen a vergrößert wird. Man erhält ein Wort, das nicht aus L ist. Äquivalent läßt sich auch im zweiten Fall des Lemmas zeigen, daß man durch Pumpen ein Wort erreicht, das nicht in L enthalten ist. □

Satz 6.5 *Sei $f \in \{t, =k, \geq k \mid k \geq 2\}$ und $m \geq 2$. Die Sprache*

$$L_m = \{(a^n b^n)^m \mid n \geq 1\}$$

ist in $\mathcal{L}$(CD-mLIN, f) aber nicht in $\mathcal{L}$(CD-$(m-1)$LIN, f).

Beweis: Das folgende CD Grammatiksystem in CD-mLIN erzeugt L_m im t-Modus. Sei $\Gamma_m = (\{S, A, B\}, \{a, b\}, P_1, \ldots, P_4, S)$ und

$$\begin{aligned} P_1 &= \{S \to A^m\}, \\ P_2 &= \{A \to aBb\}, \\ P_3 &= \{B \to aAb\}, \\ P_4 &= \{A \to ab, B \to ab\}. \end{aligned}$$

Eine Ableitung in Γ_m beginnt mit der Komponente P_1. Die Komponenten P_2 und P_3 fügen dann eine gleiche Anzahl von Symbolen a und b zu den m Teilstrings, die aus den m Nichtterminalen A bestehen, hinzu. Mit P_4 wird die Ableitung abgeschlossen.

Nach Satz 6.3 gibt es auch ein CD Grammatiksystem in (CD-mLIN, d) mit $d \in \{=k, \geq k \mid k \geq 2\}$, das L_m erzeugt.

Wir nehmen nun an, daß L_m in $\mathcal{L}$(CD-$(m-1)$LIN, f) ist. Sei h die Konstante aus Lemma 6.7. Wir wählen ein Wort w aus L_m mit $|w| > h$. Dann ist $w = (a^n b^n)^m$. Von Lemma 6.7 wissen wir aber, daß

$$w = x_1 u_1 y_1 v_1 z_1 x_2 u_2 y_2 v_2 z_2 \ldots x_{m-1} u_{m-1} y_{n-1} v_{m-1} z_{m-1}$$

gilt, und diese Zerlegung die Eigenschaft hat, daß mindestens einer der Strings u_j oder v_j nicht leer ist. Außerdem müssen u_j und v_j nur aus Symbolen a oder nur aus Symbolen b bestehen. Sonst könnte man ein Wort außerhalb von L_m erzeugen. Es gibt aber höchstens $2(m-1)$ u_j und v_j. Also werden mindestens ein und höchstens $2(m-1)$ der Teilstrings von a^n oder b^n zu $a^{n'}$ oder $b^{n'}$ erweitert. Es bleiben dann mindestens zwei Teilstrings a^n oder b^n übrig, und wir erhalten auch ein Wort, das nicht in L_m ist. Insgesamt gilt, daß $L_m \notin \mathcal{L}(CD\text{-}(m-1)LIN, f)$. □

Metalineare CD Grammatiksysteme sind laut Satz 6.3 äquivalent zu metalinearen ET0L-Systemen. Es gibt ein Pumpinglemma und eine Normalform für metalineare ET0L-Systeme aus [RV80], die natürlich auch für metalineare CD Grammatiksysteme gelten. Dieses Pumpinglemma setzt die Normalform voraus, mit der sich die Breite des ET0L-Systems aber verdoppelt. Wir haben deshalb hier zusätzliche Pumpinglemmata, die die Breite erhalten, bewiesen. Des Weiteren werden in [RV80] einige Abschlußeigenschaften bewiesen, die auch für metalineare CD Grammatiksysteme gelten. Alle Abschlußeigenschaften, die wir im folgenden benötigen, fassen wir hier zusammen und geben direkte Beweise an.

Satz 6.6 *Sei $f \in \{t, =k, \geq k \mid k \geq 2\}$ und $m \geq 1$, dann ist $\mathcal{L}(CD\text{-}mLIN, f)$ abgeschlossen unter*

1. *Vereinigung,*
2. *Schnitt mit regulären Mengen,*
3. *Homomorphismus und*
4. *inversem Homomorphismus.*

Beweis: 1. Seien L_1 und L_2 zwei Sprachen mit $L_1, L_2 \in \mathcal{L}(CD\text{-}mLIN, f)$ und $\Gamma_1 = (N_1, T_1, P_{1,1}, \ldots, P_{1,n_1}, S_1)$ und $\Gamma_2 = (N_2, T_2, P_{2,1}, \ldots, P_{2,n_2}, S_2)$ zwei m-lineare CD Grammatiksysteme mit $L(\Gamma_1, f) = L_1$ und $L(\Gamma_2, f) = L_2$. Ohne Beschränkung der Allgemeinheit können wir annehmen, daß N_1 und N_2 disjunkt sind. Sei S ein neues Symbol mit $S \notin (N_1 \cup N_2)$. Wir konstruieren für $1 \leq j_i \leq n_i$ und $1 \leq i \leq 2$ die Komponenten P'_{i,j_i} aus P_{i,j_i} wie folgt:

$$P'_{i,j_i} = \{S \to \alpha \mid S_i \to \alpha \in P_{i,j_i}\} \cup P_{i,j_i}.$$

Das m-lineare CD Grammatiksystem

$$G = (N_1 \cup N_2 \cup \{S\}, T_1 \cup T_2, P'_{1,1}, \ldots, P'_{1,n_1}, P'_{2,1}, \ldots, P'_{2,n_2}, S)$$

erzeugt $L_1 \cup L_2$ im f-Modus.

2. Sei L eine Sprache in $\mathcal{L}(CD\text{-}mLIN, f)$ und $\Gamma = (N, T, P_1, \ldots, P_n, S)$ ein CD Grammatiksystem mit $L(\Gamma, f) = L$. Weiterhin sei R eine reguläre Menge, die von einem deterministischen endlichen Automat $M = (Q, \Sigma, \delta, q_0, F)$ akzeptiert wird. Sei S' ein neues Symbol mit $S' \notin N$ und

$$N' = \{[q, A, p] \mid q, p \in Q, A \in N\} \cup \{S'\}.$$

Für alle i mit $1 \leq i \leq n$ seien $p_{i,1}, \ldots, p_{i,n_i}$ die Produktionen in P_i. Wir konstruieren jetzt für alle i mit $1 \leq i \leq n$ und alle j_i mit $1 \leq j_i \leq n_i$ die Menge

$$\begin{aligned} P_{i,j_i} \quad = \quad & \{[q,A,p] \to [q,A_1,q_1][q_1,A_2,q_2] \ldots [q_{m_0-1},A_{m_0},p] \mid \\ & q,q_1,\ldots,q_{m_0-1},p \in Q, \\ & p_{i,j_i} = A \to A_1 \ldots A_{m_0} \in P_i,\ A \in N,\ A_1,\ldots,A_{m_0} \in N \cup T\}. \end{aligned}$$

Aus P_{i,j_i} können wir nun P'_{i,j_i} erhalten, wenn wir alle Symbole $[q,a,p]$, $q,p \in Q, a \in T$ und $\delta(q,a) = p$ mit a ersetzen. Außerdem werden alle Produktionen, die Symbole $[q,a,p]$ mit $q,p \in Q, a \in T$ und $\delta(q,a) \neq p$ enthalten, aus P'_{i,j_i} entfernt. Als letztes werden die Nichtterminale $[q_0,S,q_f]$, $q_f \in F$, in allen Produktionen von P'_{i,j_i} mit S' ersetzt. Sei jetzt für $1 \leq i \leq n$ die Produktionsmenge

$$P'_i = \bigcup_{1 \leq j_i \leq n_i} P'_{i,j_i}.$$

Das m-lineare CD Grammatiksystem $\Gamma' = (N', T, P'_1, \ldots, P'_n, S')$ erzeugt nur Wörter in L, da es nur aus Produktionen besteht, die auf Produktionen von Γ basieren. Durch das Entfernen der Produktionen, die Symbole $[q,a,p]$ mit $q,p \in Q, a \in T$ und $\delta(q,a) \neq p$ enthalten, wird erreicht, daß Γ' nur Wörter in R erzeugen kann. Insgesamt ist dann $L(\Gamma', f) = L \cap R$.

3. Sei L eine Sprache in $\mathcal{L}(CD\text{-}mLIN, f)$ und $\Gamma = (N, T, P_1, \ldots, P_n, S)$ ein CD Grammatiksystem mit $L(\Gamma, f) = L$. Sei außerdem $h : T \to T'^*$ ein Homomorphismus. Mit h' erweitern wir diesen Homomorphismus auf $N \cup T$ wie folgt. Es ist $h' : N \cup T \to (N \cup T')^*$ mit $h'(a) = h(a)$ für alle $a \in T$ und $h'(A) = A$ für alle $A \in N$. Für jedes P_i mit $1 \leq i \leq n$ konstruieren wir eine Komponente $P'_i = \{A \to h'(\alpha) \mid A \to \alpha \in P_i, A \in N, \alpha \in (N \cup T)^*\}$. Das m-lineare CD Grammatiksystem $\Gamma' = (N, T', P'_1, \ldots, P'_n, S)$ generiert $h(L)$ im f-Modus.

4. Sei L eine Sprache in $\mathcal{L}(CD\text{-}mLIN, f)$. Dann gibt es nach Satz 6.3 ein CD Grammatiksystem $\Gamma = (N, T, P_1, \ldots, P_n, S)$ mit $L(\Gamma, t) = L$. Sei weiterhin $h : T' \to T^*$ ein Homomorphismus. Es kann ohne Beschränkung der Allgemeinheit angenommen werden, daß $T \cap T' = \emptyset$ ist.

Wir konstruieren jetzt eine Reihe von Sprachen, aus denen sich dann mit den vorher behandelten Operationen die gewünschte Menge $h^{-1}(L)$ ergibt. Die Sprache

$$\begin{aligned} K \quad = \quad & \{y \in (T \cup T')^* \mid y = z_0 x_1 z_1 \ldots z_{l-1} x_l z_l,\ x_1 \ldots x_l \in L(\Gamma, f), \\ & z_0, \ldots, z_l \in T'^*\} \end{aligned}$$

kann von einem CD Grammatiksystem $\Gamma' = (N', T \cup T', P'_1, \ldots, P'_n, S)$, das m-linear ist, im t-Modus erzeugt werden. Dieses CD Grammatiksystem konstruiert man ausgehend von Γ.

Seien $p_{i,1}, \ldots, p_{i,n_i}$ die Produktionen, die in P_i für $1 \leq i \leq n$ enthalten sind. Für jedes p_{i,j_i} mit $1 \leq i \leq n$ und $1 \leq j_i \leq n_i$, das in der Form

$$\begin{aligned} p_{i,j_i} : A \quad \to \quad & a_1 \ldots a_{r_1} B b_1 \ldots b_{r_2} \in P_i, \\ & A \in N \setminus \{S\},\ B \in N \cup \{\varepsilon\},\ a_1, \ldots, a_{r_1}, b_1, \ldots, b_{r_2} \in T \end{aligned}$$

ist und folglich keine Startproduktion mit mehreren Nichtterminalen auf der rechten Seite ist, konstruieren wir eine reguläre Menge

$$\begin{aligned} R_{p_{i,j_i}} &= \{z_0 a_1 z_1 \dots z_{r_1-1} a_{r_1} z_{r_1} B z_0' b_1 z_1' \dots z_{r_2-1}' b_{r_2} z_{r_2}' \mid \\ &\quad z_0, \dots, z_{r_1} \in T'^*, z_0', \dots, z_{r_2}' \in T'^*\}. \end{aligned}$$

Wir erhalten N' aus N wie folgt:

$$\begin{aligned} N' &= N \cup \{[p_{i,j_i}] \mid p_{i,j_i} : A \to a_1 \dots a_{r_1} B b_1 \dots b_{r_2} \in P_i, \\ &\quad A \in N \setminus \{S\},\ B \in N \cup \{\varepsilon\},\ a_1, \dots, a_{r_1}, b_1, \dots, b_{r_2} \in T, \\ &\quad 1 \le i \le n,\ 1 \le j_i \le n_i\}. \end{aligned}$$

Danach konstruieren wir P_i' aus P_i, indem wir jede Produktion p_{i,j_i} in P_i in der Form $A \to a_1 \dots a_{r_1} B b_1 \dots b_{r_2}$ durch $A \to [p_{i,j_i}]$ ersetzen, wobei $[p_{i,j_i}]$ eines der neuen Nichtterminale ist. Für jede reguläre Menge $R_{p_{i,j_i}}$ gibt es eine reguläre Grammatik $G_{R_{p_{i,j_i}}} = (N_{R_{p_{i,j_i}}}, T, P_{R_{p_{i,j_i}}}, [p_{i,j_i}])$. Dabei ist $N_{R_{p_{i,j_i}}} \cap N' = [p_{i,j_i}]$. Wir fügen jetzt zu P_i' alle Produktionen aus $P_{R_{p_{i,j_i}}}$ hinzu, wobei $p_{i,j_i} \in P_i$. Das jetzt resultierende CD Grammatiksystem $\Gamma' = (N', T \cup T', P_1', \dots, P_n', S)$ erzeugt K im t-Modus und ist m-linear, da nur lineare beziehungsweise reguläre Produktionen hinzugefügt wurden.

Wir betrachten nun die reguläre Menge

$$M = \{h(y_1) y_1 h(y_2) y_2 \dots h(y_n) y_n \mid n \ge 1,\ 1 \le i \le n,\ y_i \in T'\}.$$

Der Schnitt zwischen K und M kann wie folgt beschrieben werden:

$$K \cap M = \{h(y_1) y_1 h(y_2) y_2 \dots h(y_n) y_n \mid h(y_1) \dots h(y_n) \in L(\Gamma, t)\}.$$

Wenn wir jetzt den Homomorphismus $j : (T \cup T') \to T'$, $j(a) = a$ wenn $a \in T'$ und $j(a) = \varepsilon$ wenn $a \in T$, definieren, kann man sehen, daß $j(K \cap M) = h^{-1}(L(\Gamma, t))$. Da $\mathcal{L}(CD\text{-}mLIN, t)$ unter Schnitt mit regulären Mengen und Homomorphismus abgeschlossen ist, folgt, daß $h^{-1}(L) \in \mathcal{L}(CD\text{-}mLIN, t)$ ist. Wenn der Ableitungsmodus f ursprünglich nicht t war, kann man nach Satz 6.3 auch ein m-lineares CD Grammatiksystem konstruieren, das im Modus $f \in \{= k, \ge k \mid k \ge 2\}$ arbeitet und $h^{-1}(L)$ erzeugt. Insgesamt erhalten wir, daß $\mathcal{L}(CD\text{-}mLIN, f)$ unter inversem Homomorphismus abgeschlossen ist. □

Satz 6.7 *Sei $m \ge 1$ und $f \in \{t, = k, \ge k \mid k \ge 2\}$, dann ist $\mathcal{L}(CD\text{-}mLIN, f)$ nicht abgeschlossen unter*

1. *Konkatenation,*
2. *Kleenescher Hülle,*
3. *Schnitt und*
4. *Komplement.*

Beweis: 1. In Korollar 6.1 wird gezeigt, daß $L_m = \{a^n b^n \mid n \geq 0\}^{2m-2}$ in $\mathcal{L}(CD\text{-}mLIN, f)$ ist für $f \in \{t, =k, \geq k \mid k \geq 2\}$. $L_m L_m$ besteht aber aus Wörtern der Form $w = (a^h b^h)^{4m-4}$, und nach Korollar 6.1 ist die Sprache $L_m L_m$ nicht in $\mathcal{L}(CD\text{-}mLIN, f)$.

2. folgt aus 1. und der Tatsache, daß m-lineare CD Grammatiksysteme unter Schnitt mit regulären Mengen abgeschlossen sind.

3. Wir betrachten die Sprachen $L_1 = \{a^+ b^+ (a^n b^n)^m \mid n \geq 1\}$ und $L_2 = \{(a^n b^n)^m a^+ b^+ \mid n \geq 1\}$. L_1 kann im t-Modus von folgendem CD Grammatiksystem erzeugt werden:

$$
\begin{aligned}
\Gamma_1 &= \{\{S, A, B, C, F\}, \{a, b\}, P_1, P_2, P_3, P_4, P_5, S\}, \\
P_1 &= \{S \to CA^{m-1}\}, \\
P_2 &= \{C \to aC, C \to aB, B \to bB, B \to bA\}, \\
P_3 &= \{A \to aA'b, C \to F, B \to F\}, \\
P_4 &= \{A' \to A\}, \\
P_5 &= \{A \to ab, C \to F, B \to F\}.
\end{aligned}
$$

Nach Satz 6.3 gibt es auch ein m-lineares CD Grammatiksystem, das L_1 in den $\{=k, \geq k \mid k \geq 2\}$-Modi generiert. Analog können wir CD Grammatiksysteme konstruieren, die L_2 erzeugen. Der Schnitt zwischen den zwei Sprachen ist $L_1 \cap L_2 = \{(a^n b^n)^{m+1}\}$. Wir wissen aus Satz 6.5, daß diese Sprache nicht in $\mathcal{L}(CD\text{-}mLIN, f)$ ist.

4. Mit dem Satz von De Morgan und der Tatsache, daß $\mathcal{L}(CD\text{-}mLIN, f)$ unter Vereinigung, aber nicht unter Schnitt abgeschlossen ist, erhalten wir, daß $\mathcal{L}(CD\text{-}mLIN, f)$ nicht unter Komplement abgeschlossen ist. □

Lemma 6.9 *Sei $f \in \{t, =k, \geq k \mid k \geq 2\}$, dann ist $\mathcal{L}(CD\text{-}mLIN, f)$ mit $m \geq 1$ abgeschlossen unter gsm-Abbildungen.*

Beweis: Das Lemma folgt aus Lemma 2.9. □

Nach den Abschlußeigenschaften folgen nun Beweise für einige formale Eigenschaften von m-linearen CD Grammatiksystemen. Sie werden für den Beweis von Satz 6.8 gebraucht, mit dessen Hilfe wir sehen werden, daß m-lineare CD Grammatiksysteme höchstens $2m-1$ markiert konkatenierte lineare Sprachen erzeugen können.

Lemma 6.10 *Für jedes CD Grammatiksystem $\Gamma = (N, T, P_1, \ldots, P_n, S)$ in der Klasse $(CD\text{-}mLIN, f)$, wobei $m \geq 1$ und $f \in \{t, =k, \geq k \mid k \geq 2\}$ ist, gibt es ein äquivalentes CD Grammatiksystem $\Gamma' \in (CD\text{-}mLIN, f)$ mit folgenden Eigenschaften:*

1. *Jedes $A \in N$ ist der Knoten von mindestens einem gültigen Ableitungsbaum.*
2. *Jede Produktion $S \to A_1 \ldots A_{m_0}$, $m_0 \leq m$, wird an der Wurzel von mindestens einem gültigen Ableitungsbaum angewendet.*

3. *Wenn ein Nichtterminal F existiert, das zwar die Markierung eines Knotens in einem gültigen Ableitungsbaum, aber nicht die Markierung eines Knotens in einem gültigen kompletten Ableitungsbaum ist, dann kommt F nicht auf der linken Seite einer Produktion vor.*

Beweis: Wir können Γ auf folgende Weise modifizieren, um die erwünschten Eigenschaften zu erhalten. Wenn A nicht der Knoten eines gültigen Ableitungsbaums in Bezug auf f ist, wird A aus N' entfernt. Darüber hinaus werden die Produktionen in $P_1, \ldots, P_n$, die A enthalten, entfernt, da eine Ableitung, die mit S startet, diese Produktionen nicht benutzt.

Wenn es eine Startproduktion gibt, die nicht an der Wurzel eines gültigen und vollständigen Ableitungsbaums vorkommt, dann kann man diese Produktion entfernen, ohne die erzeugte Sprache zu beeinflussen.

Das Nichtterminal F blockiert eine Ableitung. Wenn es in einer Ableitung vorkommt, kann kein Terminalstring erreicht werden. Wir können also alle Produktionen, auf deren linker Seite F steht, entfernen, ohne die erzeugte Sprache zu verändern. □

Definition 6.2 *Sei T ein endliches Alphabet und $c \notin T$. Sei weiterhin $f \in \{t, = k, \geq k \mid k \geq 2\}$, $\Gamma = (N, T \cup \{c\}, P_1, \ldots, P_n, S)$ ein CD Grammatiksystem, und $L(\Gamma, f) \subseteq T^*cT^*$. Wir definieren für Γ die folgenden Mengen:*

$$
\begin{aligned}
N_{(L,f)} &= \{A \in N \mid A \text{ ist die Wurzel eines Unterbaums } t' \text{ eines} \\
&\quad \text{gültigen vollständigen Ableitungsbaums } t \text{ in Bezug auf } f \\
&\quad \text{mit } yield(t') = u \in T^* \text{ und } yield(t) = u'uvcv',\ u', v, v' \in T^*\}. \\
N_{(R,f)} &= \{A \in N \mid A \text{ ist die Wurzel eines Unterbaums } t' \text{ eines} \\
&\quad \text{gültigen vollständigen Ableitungsbaums } t \text{ in Bezug auf } f \\
&\quad \text{mit } yield(t') = u \in T^* \text{ und } yield(t) = u'cvuv',\ u', v, v' \in T^*\}. \\
N_{(c,f)} &= \{A \in N \mid A \text{ ist die Wurzel eines Unterbaums } t' \\
&\quad \text{eines vollständigen gültigen Ableitungsbaums in Bezug auf } f \\
&\quad \text{und } yield(t') = u_1cu_2, u_1, u_2 \in T^*\}. \\
N_{(B,f)} &= \{A \in N \mid A \text{ ist der Knoten eines gültigen Ableitungsbaums in} \\
&\quad \text{Bezug auf } f, \text{ aber } A \text{ ist nicht der Knoten eines vollständigen} \\
&\quad \text{gültigen Ableitungsbaums in Bezug auf } f\}.
\end{aligned}
$$

Lemma 6.11 *Sei T ein endliches Alphabet und $c \notin T$. Sei weiterhin $L \subseteq T^*cT^*$ eine Sprache in $\mathcal{L}$(CD-mLIN, f), wobei $m \geq 2$ und $f \in \{t, = k, \geq k \mid k \geq 2\}$. Dann kann man ein CD Grammatiksystem $\Gamma = (N, T, P_1, \ldots, P_n, S) \in$ (CD-mLIN, f) effektiv konstruieren, das L erzeugt und die Eigenschaft hat, daß $\{N_{(L,f)}, N_{(c,f)}, N_{(R,f)}, N_{(B,f)}\}$ eine Partition von N ist.*

Beweis: Sei $\Gamma' = (N', T, P'_1, \ldots, P'_n, S)$ ein CD Grammatiksystem mit $\Gamma' \in$ (CD-mLIN, f), $f \in \{t, = k, \geq k \mid k \geq 2\}$ und $L(\Gamma', f) = L$. Wir können annehmen, daß Γ' entsprechend Lemma 6.10 aufgebaut ist. Weiterhin sei die Menge $\{N'_{(L,f)}, N'_{(c,f)}, N'_{(R,f)}, N'_{(B,f)}\}$ definiert wie $\{N_{(L,f)}, N_{(c,f)}, N_{(R,f)}, N_{(B,f)}\}$ in Definition 6.2. Da die Nichtterminale in $N'_{(B,f)}$ in keinem vollständigen gültigen Ableitungsbaum in Bezug auf f vorkommen, folgt, daß $N'_{(B,f)} \cap (N'_{(L,f)} \cup N'_{(R,f)} \cup N'_{(c,f)}) = \emptyset$.

Wir werden jetzt ein CD Grammatiksystem $\Gamma = (N, T, P_1, \ldots, P_n, S_c)$ konstruieren, das L im f-Modus erzeugt und bei dem $\{N_{(L,f)}, N_{(c,f)}, N_{(R,f)}, N_{(B,f)}\}$ eine Partition von N ist. Es gilt

$$N = N'_{(B,f)} \cup \{A_R \mid A \in N'_{(R,f)}\} \cup \{A_L \mid A \in N'_{(L,f)}\} \cup \{A_c \mid A \in N'_{(c,f)}\}.$$

Für $1 \leq i \leq n$ definieren wir

$$\begin{aligned}
P_i \;=\; & \{A_R \to uB_Rv \mid A \to uBv \in P'_i,\ A, B \in N'_{(R,f)},\ u, v \in T^*\} \cup \\
& \{A_L \to uB_Lv \mid A \to uBv \in P'_i,\ A, B \in N'_{(L,f)},\ u, v \in T^*\} \cup \\
& \{A_c \to uB_cv \mid A \to uBv \in P'_i,\ A, B \in N'_{(c,f)},\ u, v \in T^*\} \cup \\
& \{A_c \to uB_Lvcv' \mid A \to uBvcv' \in P'_i,\ A \in N'_{(c,f)}, B \in N'_{(L,f)}, \\
& \ u, v, v' \in T^*\} \cup \\
& \{A_c \to ucu'B_Rv \mid A \to ucu'Bv' \in P'_i,\ A \in N'_{(c,f)}, B \in N'_{(R,f)}, \\
& \ u, u', v \in T^*\} \cup \\
& \{A_x \to uFv \mid A \to uFv \in P'_i,\ A \in N'_x,\ x \in \{R, L, c\},\ F \in N'_{(B,f)}, \\
& \ u, v \in T^*cT^* \cup T^*\} \cup \\
& \{S_c \to A_{1\,L} \ldots A_{j-1\,L} A_{j\,c} A_{j+1\,R} \ldots A_{m_0\,R} \mid S \to A_1 \ldots A_{m_0} \in P'_i, \\
& \ A_1, \ldots, A_{j-1} \in N'_{(L,f)},\ A_j \in N'_{(c,f)},\ A_{j+1}, \ldots, A_{m_0} \in N'_{(R,f)}\}.
\end{aligned}$$

Wir betrachten nun $\{N_{(B,f)}, N_{(L,f)}, N_{(R,f)}, N_{(c,f)}\}$ wie in Definition 6.2 gegeben. Da die Nichtterminale in Γ mit R, L oder c markiert sind, je nach dem, ob sie einen String rechts von einem c, links von einem c oder mit einem c generieren, ist die Menge $\{N_{(B,f)}, N_{(L,f)}, N_{(R,f)}, N_{(c,f)}\}$ eine Partition von N. Γ besteht aus Produktionen von Γ', deren Nichtterminale lediglich mit L, R oder c markiert sind. Es gilt also $L(\Gamma, f) \subseteq L(\Gamma', f)$. Die Ableitungen in Γ verlaufen genauso wie in Γ'. Der Unterschied ist, daß Nichtterminale, die einen String links beziehungsweise rechts eines Symbols c erzeugen entsprechend mit L oder R markiert sind, und Nichtterminale, die einen String erzeugen, der c beinhaltet, mit c markiert sind. Der Unterschied zwischen den Ableitungen beider Grammatiksysteme besteht in Ableitungen, die keinen Terminalstring erzeugen. Durch das Entfernen von Produktionen, die auf der linken Seite ein Nichtterminal haben, das keinen Terminalstring erzeugen kann, ist es möglich, daß Ableitungen, die in Γ' blockiert werden, in Γ schon früher blockiert sind. Das verändert aber die erzeugte Sprache nicht. Es ist also auch $L(\Gamma', f) \subseteq L(\Gamma, f)$. □

Wir können jetzt einen Satz über die Eigenschaften von Sprachen, die von metalinearen CD Grammatiksystemen erzeugt werden, beweisen. Ähnliche Ergebnisse gibt es für metalineare Sprachen ([Gre66]). Der im folgenden gezeigte Satz kann für die Trennung von Sprachen in $\mathcal{L}(\textit{CD-}(m+1)\textit{LIN}, f)$ von Sprachen in $\mathcal{L}(\textit{CD-}m\textit{LIN}, f)$ benutzt werden. Außerdem wird er im nächsten Abschnitt ein wichtiges Hilfsmittel für die Beweise der Ergebnisse in der Beschreibungskomplexität sein.

Satz 6.8 *Sei $f \in \{t, =k, \geq k \mid k \geq 2\}$, T ein endliches Alphabet mit $c \notin T$ und seien $L_1 \subseteq T^*$, $L_2 \subseteq T^*$ mit*

1. *$L_1 \in \mathcal{L}(CD\text{-}mLIN, f) \setminus \mathcal{L}(CD\text{-}(m-1)LIN, f)$ mit $m \geq 2$ oder $L_1 \in \mathcal{L}(LIN) \setminus \mathcal{L}(REG)$ und*
2. *$L_2 \in \mathcal{L}(2LIN) \setminus \mathcal{L}(LIN)$.*

Es gilt für $m \geq 1$

$$L_1 c L_2 \in \mathcal{L}(CD\text{-}(m+1)LIN, f) \setminus \mathcal{L}(CD\text{-}mLIN, f).$$

Beweis: Wir zeigen zuerst, daß man ein CD Grammatiksystem Γ aus der Klasse $(CD\text{-}(m+1)LIN, f)$ konstruieren kann, das $L_1 c L_2$ erzeugt. Wir benutzen dabei ein m-lineares CD Grammatiksystem für L_1 und eine 2-lineare kontextfreie Grammatik für L_2.

Sei $L_1 \in \mathcal{L}(CD\text{-}mLIN, f)$ und $L_2 \in \mathcal{L}(2LIN)$, dann gibt es nach Satz 6.3 ein CD Grammatiksystem $\Gamma_1 \in (CD\text{-}mLIN, t)$ und eine 2-lineare Grammatik G_2 mit $L(\Gamma_1, t) = L_1$ und $L(G_2) = L_2$.

Seien $s_1, \ldots, s_{p_1}$ die Startproduktionen von Γ_1 und $r_1, \ldots, r_{p_2}$ die Startproduktionen von G_2. Wir können annehmen, daß Γ_1 und G_2 disjunkte Mengen von Nichtterminalen haben.

Seien jetzt die CD Grammatiksysteme $\Gamma_1^{s_{i_0}}$ konstruiert aus Γ_1, indem alle Startproduktionen s_i mit $1 \leq i \neq i_0 \leq p_1$ entfernt wurden. Seien weiterhin die Grammatiken $G_2^{r_{j_0}}$ aus G_2 konstruiert, indem die Startproduktionen r_j mit $1 \leq j \neq j_0 \leq p_2$ entfernt wurden. Das bedeutet also, daß die einzige Startproduktion in $\Gamma_1^{s_{i_0}}$ die Produktion s_{i_0} ist. Darüber hinaus ist die einzige Startproduktion in $G_2^{r_{j_0}}$ die Produktion r_{j_0}. Man kann sehen, daß $L_1 = \bigcup_{i=1}^{p_1} L(\Gamma_1^{s_i}, t)$ und $L_2 = \bigcup_{i=1}^{p_2} L(G_2^{r_i})$. Es folgt, daß

$$L_1 c L_2 = \bigcup_{i=1}^{p_1} \bigcup_{j=1}^{p_2} L(\Gamma_1^{s_i}, t) c L(G_2^{r_j}).$$

Wir konstruieren jetzt CD Grammatiksysteme für $L(\Gamma_1^{s_i}, t) c L(G_2^{r_j})$.

Wenn r_j eine lineare Produktion ist, dann ist $L(G_2^{r_j})$ eine lineare Sprache, und es existiert ein CD Grammatiksystem $\Gamma_2^{r_j} \in (CD\text{-}1LIN, t)$ mit $L(\Gamma_2^{r_j}, t) = L(G_2^{r_j})$, das nur eine Startproduktion r_j hat. Wir können annehmen, daß die Nichtterminale von Γ_1 und $\Gamma_2^{r_j}$ disjunkt sind. Sei $s_i = S \to A_1 \ldots A_{m_0}$, $m_0 \leq m$, S_2 das Startsymbol von $\Gamma_2^{r_j}$ und $r_j = S_2 \to \alpha$, wobei α höchstens ein Nichtterminal enthält. Wir erhalten $\Gamma_1^{'s_i}$ aus $\Gamma_1^{s_i}$ wie folgt. Wir ersetzen s_i durch $S \to A_1 \ldots A_{m_0} S_2$. Die Menge der Nichtterminale von $\Gamma_1^{'s_i}$besteht aus den Nichtterminalen von $\Gamma_2^{r_j}$ und $\Gamma_1^{s_i}$. Die Komponenten von $\Gamma_1^{'s_i}$ sind die Komponenten von $\Gamma_1^{s_i}$ ergänzt durch die Komponenten von $\Gamma_2^{r_j}$. Wir ersetzen dann noch jedes Vorkommen von r_j durch $S_2 \to c\alpha$. Die Terminale von $\Gamma_1^{'s_i}$ sind die Terminale von $\Gamma_1^{s_i}$ vereinigt mit den Terminalen von $\Gamma_2^{r_j}$ und dem Symbol $\{c\}$. Da die Mengen der Nichtterminale von $\Gamma_1^{s_i}$ und $\Gamma_2^{r_j}$ disjunkt sind, erzeugt das neue

Grammatiksystem ein Wort aus $L(\Gamma_1^{s_i}, t)$ und ein Wort aus $L(\Gamma_2^{r_j}, t)$. Durch die veränderte Produktion r_j sind beide Wörter durch das Symbol c getrennt. Da die Produktionen von $\Gamma_2^{r_j}$ linear sind, ist das gesamte CD Grammatiksystem $\Gamma_1'^{s_i}$ $(m+1)$-linear und $L(\Gamma_1'^{s_i}, t) = L(\Gamma_1^{s_i}, t)cL(\Gamma_2^{r_j}, t) = L(\Gamma_1^{s_i}, t)cL(G_2^{r_j})$.

Sei jetzt $r_j = S \to A_1B_2$. Seien weiterhin $G^{(1)} = (N^{(1)}, T, P^{(1)}, A^{(1)})$ und $G^{(2)} = (N^{(2)}, T, P^{(2)}, B^{(2)})$ zwei von den Nichtterminalen her disjunkte Kopien von $G_2^{r_j}$, wobei S durch $A^{(1)}$ beziehungsweise $B^{(2)}$ ersetzt wurde. Die Homomorphismen (1) und (2) sind wie im 2. Kapitel definiert. Außerdem wurde bei beiden Kopien die Produktion r_j entfernt, was beide Grammatiken zu linearen Grammatiken macht.

Wir können sehen, daß $N^{(1)} \cap N^{(2)} = \emptyset$ gilt. Außerdem kann man nach der Normalform für lineare Grammatiken aus Lemma 2.3 annehmen, daß die Produktionen in der Form $C \to aD, C \to Da$ oder $C \to a$ sind, wobei a ein einzelnes Nichtterminal ist und $A^{(1)}$ und $B^{(2)}$ nicht auf der rechten Seite von einer Produktion vorkommen. Da r_j die einzige Startproduktion in $G_2^{r_j}$ ist, gilt $L(G_2^{r_j}) = L(G^{(1)})L(G^{(2)})$.

Wenn jetzt $\Gamma_1^{s_i} = (N, T_1, P_1', \ldots, P_n', S)$ und $s_i = S \to E_1 \ldots E_{m_0}$ ist, wobei $E_1, \ldots, E_{m_0} \in N$ gilt, können wir mit Hilfe der zwei linearen Grammatiken $G^{(1)}$ und $G^{(2)}$ ein CD Grammatiksystem konstruieren, das im t-Modus arbeitet und $L(\Gamma_1^{s_i}, t)cL(G_2^{r_j})$ erzeugt. Dabei können wir annehmen, daß N und $N^{(1)}$ beziehungsweise N und $N^{(2)}$ disjunkt sind. Darüber hinaus seien $p_1, \ldots, p_z$ die Produktionen in $P^{(1)}$. Wir betrachten das folgende CD Grammatiksystem:

$$\begin{aligned}\Gamma_1'^{s_i} \;=\; & (N \cup N' \cup \{F\} \cup N^{(1)} \cup N'^{(1)} \cup N''^{(1)} \cup N'''^{(1)} \cup N^{(2)}, T_1 \cup T_2 \cup \{c\},\\ & P_1', \ldots, P_n', P_{n+1}', \ldots, P_{n+2+z}', S),\end{aligned}$$

wobei s_i durch $s_i' = S \to E_1' \ldots E_{m_0-1}' A^{(1)} A''^{(1)}$ ersetzt wurde. Die Markierungen $'$ an den Nichtterminalmengen bedeuten, daß alle enthaltenen Nichtterminale mit der entsprechenden Anzahl von Symbolen $'$ markiert sind. Es ist $F \notin N \cup N' \cup N^{(1)} \cup N'^{(1)} \cup N''^{(1)} \cup N'''^{(1)} \cup N^{(2)}$ und außerdem sind

$$\begin{aligned}P_{n+1}' \;=\; & P^{(2)}\\ P_{n+1+q}' \;=\; & \{C^{(1)} \to D'^{(1)}a, C''^{(1)} \to D'''^{(1)}\} \text{ wenn } p_q = D^{(1)} \to aC^{(1)},\\ & \{C^{(1)} \to D'^{(1)}, C''^{(1)} \to aD'''^{(1)}\} \text{ wenn } p_q = D^{(1)} \to C^{(1)}a,\\ & \{A^{(1)} \to C'^{(1)}a, A''^{(1)} \to C'''^{(1)}\} \text{ wenn } p_q = C^{(1)} \to a,\\ & \{C^{(1)} \to E_{m_0}ca, C''^{(1)} \to B_2, E_1' \to E_1, \ldots, E_{m_0-1}' \to E_{m_0-1}\} \cup\\ & \{H \to F \mid H \in N^{(1)} \setminus \{C^{(1)}\}\} \text{ wenn } p_q = A^{(1)} \to aC^{(1)},\\ & \{C^{(1)} \to E_{m_0}c, C''^{(1)} \to aB^{(2)}, E_1' \to E_1, \ldots, E_{m_0-1}' \to E_{m_0-1}\} \cup\\ & \{H \to F \mid H \in N^{(1)} \setminus \{C^{(1)}\}\} \text{ wenn } p_q = A^{(1)} \to C^{(1)}a,\\ & 1 \le q \le z,\\ P_{n+2+z}' \;=\; & \{K' \to K, K''' \to K'' \mid K \in N^{(1)}\}.\end{aligned}$$

$\Gamma_1^{'s_i}$ arbeitet im t-Modus, und jede Ableitung beginnt mit s_i'. Danach können nur die Komponenten P'_{n+1+q} und P_{n+2+z} abwechselnd an der Satzform arbeiten. Sie simulieren eine Ableitung von $G^{(1)}$, die aus Ableitungsbäumen mit einem Pfad besteht. Wir werden den am weitesten links stehenden Pfad des Ableitungsbaums von $\Gamma_1^{'s_i}$ für die Simulation benutzen. Jede Produktion $C \to aD$ von $G^{(1)}$ wird simuliert durch die Produktionen $D \to C'a$ und $D'' \to C'''$. Jede Produktion $C \to Da$ von $G^{(1)}$ wird von den zwei Produktionen $D \to C'$ und $D'' \to aC'''$ simuliert. Wenn die Produktionen in $\Gamma_1^{'s_i}$ in der gleichen Reihenfolge wie in $G^{(1)}$ benutzt würden, würden die erzeugten Wörter so verändert, daß die im Original am weitesten in der Mitte stehenden Terminale in der Simulation außen stünden. Es würden also die Terminale, die im letzten Schritt einer Ableitung von $G^{(1)}$ erzeugt werden, in der Mitte des Wortes aus $L(G^{(1)})$ stehen, aber an den Rändern des Wortes, das von $\Gamma_1^{'s_i}$ erzeugt wird.

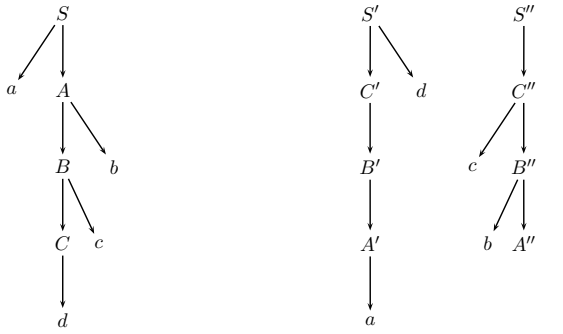

Abbildung 6.1: Erzeugung eines Wortes mit einem linearen Ableitungsbaum bzw. zwischen zwei linearen Ableitungsbäumen

Deshalb werden die Produktionen von $G^{(1)}$ in $\Gamma_1^{'s_i}$ in umgekehrter Reihenfolge ausgeführt. Das Prinzip wird in Abbildung 6.1 verdeutlicht. Sobald die Simulation einer Ableitung aus $G^{(1)}$ beendet wurde, werden alle Nichtterminale E_i' durch E_i ersetzt. Mit den Produktionen $H \to F$ wird sichergestellt, daß das nur nach der Simulation einer Ableitung von $G^{(1)}$ passieren kann. Außerdem wird das am weitesten rechts in der Ableitung stehende Nichtterminal durch $B^{(2)}$ ersetzt. Mit P'_{n+1} kann jetzt ein Wort aus $L(G^{(2)})$ erzeugt werden und mit den Komponenten $P'_1, \ldots, P'_n$ ein Wort aus $L(\Gamma_1^{s_i}, t)$. Also ist $L(\Gamma_1^{'s_i}, t) = L(\Gamma_1^{s_i}, t)cL(G_2^{r_j})$.

Wenn für den ursprünglichen Ableitungsmodus f gilt, daß $f \in \{= k, \geq k\}$ mit $k \geq 2$ ist, dann kann man das eben konstruierte Grammatiksystem, das L_1cL_2 im t-Modus erzeugt und in (CD-$(m+1)LIN, t$) ist, wie folgt umwandeln: Wir wenden Satz 6.6 an und bekommen ein CD Grammatiksystem in (CD-$(m+1)LIN, f$), das L_1cL_2 erzeugt. Somit erhalten wir für alle Modi $f \in \{t, = k, \geq k \mid k \geq 2\}$, daß $L_1cL_2 \in \mathcal{L}(CD\text{-}(m+1)LIN, f)$ ist.

Wir nehmen jetzt an, daß $L_1cL_2 \in \mathcal{L}(CD\text{-}mLIN, f)$, was zu einem Widerspruch führen wird. Sei also $\Gamma = (N, T, P_1, \ldots, P_n, S) \in (CD\text{-}mLIN, f)$ und habe Γ die Eigenschaften aus Lemma 6.10 und Lemma 6.11. Sei weiterhin $\{N_{(L,f)}, N_{(c,f)}, N_{(R,f)}, N_{(B,f)}\}$ eine Partition von N und $L(\Gamma, f) = L_1cL_2$. Für

$1 \leq i \leq n$ sei P_i' die Untermenge von P_i, die aus den folgenden Produktionen besteht:

1. $A \to \alpha$, $A \to \alpha$ ist in P_i und $A \neq S$,
2. $S \to A_1 \dots A_{m_0}$, $S \to A_1 \dots A_{m_0} \in P_i$, $m_0 = m$ und $A_{m_0} \in N_{(c,f)}$.

Für $1 \leq i \leq n$ sei P_i'' die Untermenge von P_i, die aus den folgenden Produktionen besteht:

1. $A \to \alpha$, $A \to \alpha$ ist in P_i und $A \neq S$,
2. $S \to A_1 \dots A_{m_0}$, $S \to A_1 \dots A_{m_0} \in P_i$, $m_0 < m$ oder $A_{m_0} \notin N_{(c,f)}$.

Seien $\Gamma' = (N, T, P_1', \dots, P_n', S)$ und $\Gamma'' = (N, T, P_1'', \dots, P_n'', S)$. Seien weiterhin $L' = L(\Gamma', f)$ und $L'' = L(\Gamma'', f)$. Es sei bemerkt, daß jede Produktion aus P_i entweder zu P_i' oder zu P_i'' oder zu beiden Mengen gehört. Wir erhalten, daß $L' \cup L'' \subseteq L_1 c L_2$. Da jeder gültige Ableitungsbaum t' von Γ in Bezug auf f entweder nur aus Produktionen in $\bigcup_{i=1}^{n} P_i'$ oder nur aus Produktionen in $\bigcup_{i=1}^{n} P_i''$ besteht, gilt sogar $L' \cup L'' = L_1 c L_2$.

Seien μ und ν zwei gsm-Abbildungen von $T^* c T^*$ auf T^*, die definiert sind durch $\mu(xcy) = x$ und $\nu(xcy) = y$. Dann ist $\mu(L_1 c L_2) = L_1$ und $\nu(L_1 c L_2) = L_2$. Wir werden jetzt zwei CD Grammatiksysteme konstruieren, die $\mu(L'')$ und $\nu(L')$ erzeugen. Danach werden wir zeigen, daß $\nu(L') \in \mathcal{L}(LIN)$, was zu einem Widerspruch führt.

Da wir mit Grammatiksystemen arbeiten, ist es nicht möglich, einfach die Produktionen und Nichtterminale von Γ' oder Γ'', die nicht zur Erzeugung von $\mu(L'')$ oder $\nu(L')$ benutzt werden, zu entfernen. Wir werden deshalb zwei Schritte durchführen.

Zuerst werden wir aus allen Produktionen die Terminale entfernen, die nicht für die Erzeugung von $\mu(L'')$ oder $\nu(L')$ gebraucht werden. Dadurch erhalten wir zwei CD Grammatiksysteme $\Gamma'^{(1)}$ und $\Gamma'^{(2)}$ mit $\mu(L'') = L(\Gamma'^{(1)}, f)$ und $\nu(L') = L(\Gamma'^{(2)}, f)$. In den Ableitungsbäumen von $\Gamma'^{(1)}$ und $\Gamma'^{(2)}$ können dann Nichtterminale, die keine Terminale erzeugen, zusammengefaßt werden. Dadurch erhalten wir die CD Grammatiksysteme $\Gamma''^{(1)}$ und $\Gamma''^{(2)}$, für die $\mu(L'') = L(\Gamma''^{(1)}, f)$ und $\nu(L') = L(\Gamma''^{(2)}, f)$ gilt.

In einem ersten Schritt werden die Nichtterminale, die nicht für $\mu(L'')$ und $\nu(L')$ gebraucht werden, entfernt. Sei $P_i'^{(1)}$ mit $1 \leq i \leq n$ die Untermenge von P_i'', die aus folgenden Produktionen besteht:

1. $A \to \alpha$, $A \to \alpha$ ist in P_i'' und $A \in N_{(L,f)}$,
2. $A \to B$, $A \to uBv$ ist in P_i'' und $A \in N_{(R,f)}$, $u, v \in T^*$, $B \in N_{(B,f)} \cup N_{(R,f)} \cup \{\varepsilon\}$,
3. $A \to uB$, $A \to uBv$ ist in P_i'' und $A \in N_{(c,f)}$, $u, v \in T^*$,
4. $A \to u$, $A \to ucv$ ist in P_i'' und $A \in N_{(c,f)}$, $u, v \in T^*$,
5. $A \to uB$, $A \to ucu'Bv$ ist in P_i'' und $A \in N_{(c,f)}$, $u, u', v \in T^*$,

6. $A \to uBv$, $A \to uBvcv'$ ist in P_i'' und $A \in N_{(c,f)}$, $u, v, v' \in T^*$,
7. $S \to A_1 \dots A_{m_0}$, $S \to A_1 \dots A_{m_0} \in P_i''$.

Sei $\Gamma'^{(1)} = (N, T, P_1'^{(1)}, \dots, P_n'^{(1)}, S)$, dann ist $\mu(L'') = L(\Gamma'^{(1)}, f)$.

Sei $P_i'^{(2)}$ mit $1 \leq i \leq n$ die Untermenge von P_i', die aus folgenden Produktionen besteht:

1. $A \to \alpha$, $A \to \alpha$ ist in P_i' und $A \in N_{(R,f)}$,
2. $A \to B$, $A \to uBv$ ist in P_i' und $A \in N_{(L,f)}$, $u, v \in T^*$, $B \in N_{(B,f)} \cup N_{(L,f)} \cup \{\varepsilon\}$,
3. $A \to Bv$, $A \to uBv$ ist in P_i' und $A \in N_{(c,f)}$, $u, v \in T^*$,
4. $A \to v$, $A \to ucv$ ist in P_i' und $A \in N_{(c,f)}$, $u, v \in T^*$,
5. $A \to u'Bv$, $A \to ucu'Bv$ ist in P_i' und $A \in N_{(c,f)}$, $u, u', v \in T^*$,
6. $A \to Bv'$, $A \to uBvcv'$ ist in P_i' und $A \in N_{(c,f)}$, $u, v, v' \in T^*$,
7. $S \to A_1 \dots A_{m_0}$, $S \to A_1 \dots A_{m_0} \in P_i'$.

Sei weiterhin $\Gamma'^{(2)} = (N, T, P_1'^{(2)}, \dots, P_n'^{(2)}, S)$, dann ist $\nu(L') = L(\Gamma'^{(2)}, f)$.

In den Ableitungsbäumen von $\Gamma'^{(1)}$ und $\Gamma'^{(2)}$ werden jetzt die Pfade ohne Terminale zu einem Pfad zusammengefaßt.

Sei $N''^{(2)} = \{(B_1, \dots, B_{m-1}, A) \mid B_1, \dots, B_{m-1} \in N_{(L,f)} \cup \{\varepsilon\},\ A \in N_{(c,f)} \cup N_{(R,f)} \cup \{\varepsilon\}\}$. Sei weiterhin $P_0''^{(2)} = \{(\varepsilon, \dots, \varepsilon) \to (\varepsilon, \dots, \varepsilon),\ (\varepsilon, \dots, \varepsilon) \to \varepsilon\}$, wobei $(\varepsilon, \dots, \varepsilon)$ ein m-dimensionaler Vektor aus leeren Worten ist. Die Produktionsmenge $P_i''^{(2)}$ besteht aus den folgenden Produktionen:

1. $S \to (A_1, \dots, A_m),\ S \to A_1 \dots A_m \in P_i'^{(2)}$,
2. $(A_1, \dots, A_m) \to (A_1, \dots, A_{j-1}, \beta, A_{j+1}, \dots, A_{m-1}, A_m),\ 1 \leq j \leq m-1,$ $A_j \to \beta \in P_i'^{(2)},\ A_j \in N_{(L,f)},\ \beta \in N_{(L,f)} \cup \{\varepsilon\}$,
3. $(A_1, \dots, A_m) \to u(A_1, \dots, A_{m-1}, B)v, A_m \to uBv \in P_i'^{(2)}, A_m \in N_{(c,f)} \cup N_{(R,f)}, u, v \in (T \cup \{c\})^*, B \in N_{(c,f)} \cup N_{(R,f)} \cup \varepsilon$.

Sei $\Gamma''^{(2)} = (N''^{(2)}, T, P_0''^{(2)}, P_1''^{(2)}, \dots, P_n''^{(2)}, S)$. Da nur der am weitesten rechts stehende Pfad eines gültigen Ableitungsbaums von $\Gamma'^{(2)}$ in Bezug auf den Ableitungsmodus f Terminale erzeugen kann, gibt es genau dann einen gültigen Ableitungsbaum in $\Gamma''^{(2)}$ mit Blattfolge $u(A_1, \dots, A_m)v$, wenn es einen gültigen Ableitungsbaum in $\Gamma'^{(2)}$ mit Blattfolge $A_1 \dots A_{m-1}uA_mv$ gibt. Es folgt, daß $\nu(L') = L(\Gamma''^{(2)}, f)$ ist, und es sei weiterhin bemerkt, daß $\Gamma''^{(2)} \in (CD\text{-}1LIN, f)$ gilt.

Sei $N''^{(1)} = \{(A, B_1, \dots, B_{m-1}) \mid B_1, \dots, B_{m-1} \in N_{(R,f)} \cup \{\varepsilon\},\ A \in N_{(c,f)} \cup N_{(L,f)} \cup \{\varepsilon\}\}$. Sei weiterhin $P_0''^{(1)} = \{(\varepsilon, \dots, \varepsilon) \to (\varepsilon, \dots, \varepsilon),\ (\varepsilon, \dots, \varepsilon) \to \varepsilon\}$, wobei $(\varepsilon, \dots, \varepsilon)$ ein m-dimensionaler Vektor von leeren Worten ist. $P_i''^{(1)}$ besteht aus den folgenden Produktionen:

1. $S \to A_1 \dots A_{m_1}(A, B_1, \dots B_{m_2}, \varepsilon, \dots, \varepsilon),\ \ S \to A_1 \dots A_{m_1} A B_1 \dots B_{m_2}, \in P_i'^{(1)}, A \in N_{(c,f)}$, $m_1 + 1 < m, m_1 + m_2 + 1 \leq m$,
2. $(B_1, \dots, B_m) \to (B_1, \dots, B_{j-1}, \beta, B_{j+1}, \dots, B_{m-1}, B_m),\ 2 \leq j \leq m$, $B_j \to \beta \in P_i'^{(1)},\ B_j \in N_{(R,f)},\ \beta \in N_{(R,f)} \cup \{\varepsilon\}$,
3. $(B_1, \dots, B_m) \to u(C, B_2 \dots, B_m)v, B_1 \to uCv \in P_i'^{(1)}, B_1 \in N_{(c,f)} \cup N_{(L,f)}$, $u, v \in T^*, C \in N_{(c,f)} \cup N_{(L,f)} \cup \varepsilon$
4. $A \to \alpha,\ A \to \alpha \in P_i'^{(1)}\ A \in N_{(L,f)}$.

Sei $\Gamma''^{(1)} = (N''^{(1)}, T, P_0''^{(1)}, P_1''^{(1)}, \dots, P_n''^{(1)}, S)$. In 1. werden die Startproduktionen in zwei Teile eingeteilt. Die Nichtterminale in $N_{(L,f)}$ bleiben unverändert, und die Nichtterminale in $N_{(c,f)} \cup N_{(R,f)}$ werden zu einem Nichtterminal zusammengefaßt. Es sei bemerkt, daß in einer Startproduktion höchstens $m-2$ Nichtterminale zu $N_{(L,f)}$ und ein Nichtterminal zu $N_{(c,f)}$ gehören. Da nur Nichtterminale in $N_{(c,f)}$ oder $N_{(L,f)}$ Terminale erzeugen können, gibt es einen gültigen Ableitungsbaum von $\Gamma''^{(1)}$ in Bezug zum Ableitungsmodus f mit der Blattfolge

$$v_0 A_1 v_1 A_2 v_2 \dots A_{m_1} v_{m_1} u(A, B_1, \dots, B_{m-1})v$$

genau dann, wenn es einen gültigen Ableitungsbaum von $\Gamma'^{(1)}$ mit der Blattfolge

$$v_0 A_1 v_1 A_2 v_2 \dots A_{m_1} v_{m_1} u A v B_1 \dots B_{m-1}$$

gibt. Daraus folgt, daß $\mu(L'') = L(\Gamma''^{(1)}, f)$. Weiterhin sei bemerkt, daß $m_1 + 1 < m$ in 1. und deshalb $\Gamma''^{(1)} \in (CD\text{-}(m-1)LIN, f)$.

Wir haben gesehen, daß $\Gamma''^{(1)}$ nur in $CD\text{-}(m-1)LIN$ ist. Daher gibt es ein Wort $w_1 \in \mu(L') \setminus \mu(L'')$, und man kann sehen, daß $w_1 c L_2 \subseteq L'$ und deshalb $L_2 = \nu(L')$. Allerdings ist $\nu(L')$ nur in $L(CD\text{-}1LIN, f)$, also in $\mathcal{L}(LIN)$. Wir erhalten einen Widerspruch zu der Voraussetzung, daß $L_2 \in \mathcal{L}(2LIN) \setminus \mathcal{L}(LIN)$ ist. □

Korollar 6.1 *Sei $f \in \{t, =k, \geq k \mid k \geq 2\}$ und $m \geq 2$. Weiterhin sei die Sprache $L \subseteq T^* \in \mathcal{L}(LIN) \setminus \mathcal{L}(REG)$ und $c \notin T$, dann gilt*

$$(Lc)^{2m-2} L \in \mathcal{L}(CD\text{-}mLIN, f) \setminus \mathcal{L}(CD\text{-}(m-1)LIN, f).$$

Beweis: Laut Lemma 2.4 ist LcL in $\mathcal{L}(2LIN) \setminus \mathcal{L}(LIN)$. Seien $c_1, \dots, c_{m-1} \notin (T \cup \{c\})$ paarweise verschiedene Symbole. Mit Induktion über die Anzahl der Symbole und dem Ergebnis aus Satz 6.8 kann man zeigen, daß

$$Lc_1 LcLc_2 \dots cLc_{m-1} LcL \in \mathcal{L}(CD\text{-}mLIN, f) \setminus \mathcal{L}(CD\text{-}(m-1)LIN, f).$$

Als nächstes konstruiert man eine gsm-Abbildung, die die Sprache $(Lc)^{2m-2}L$ auf die Sprache $Lc_1 LcLc_2 \dots cLc_{m-1} LcL$ abbildet. Sie liest die Eingaben und bildet alle Terminale aus T auf sich selbst ab. Das erste c wird auf c_1, das zweite c auf c, das dritte c auf c_2 und das vierte c auf c abgebildet und so fort.

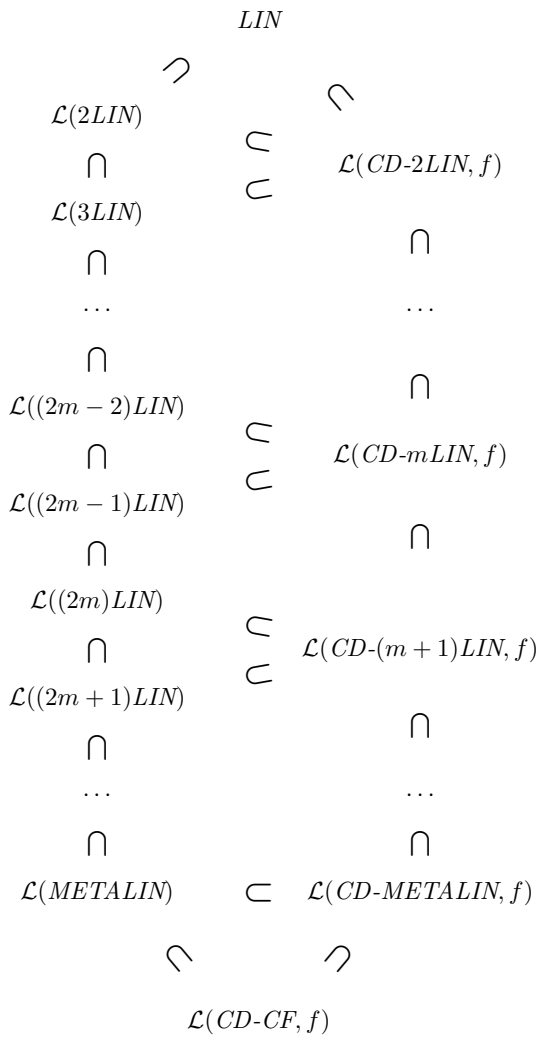

Abbildung 6.2: Die generative Mächtigkeit von metalinearen CD Grammatiksystemen

Weiterhin gibt es auch eine gsm-Abbildung, die $Lc_1LcLc_2 \dots cLc_{m-1}LcL$ auf $(Lc)^{2m-2}L$ abbildet. Sie liest die Eingaben und bildet alle Terminale aus T auf sich selbst ab. Außerdem werden $c, c_1 \dots c_{m-1}$ auf c abgebildet.

Da $\mathcal{L}(CD\text{-}mLIN, f)$ unter gsm-Abbildungen abgeschlossen ist und man beide betrachteten Sprachen aufeinander abbilden kann, gilt auch, daß $(Lc)^{2m-2}L \in \mathcal{L}(CD\text{-}mLIN, f) \setminus \mathcal{L}(CD\text{-}(m-1)LIN, f)$ ist. □

In Abbildung 6.2 kann man sehen, welche Hierarchien von Sprachklassen sich durch Korollar 6.1 und Lemma 2.6 ergeben.

6.3 Beschreibungskomplexität

Nach dem Abschnitt über die generative Mächtigkeit von metalinearen CD Grammatiksystemen werden wir jetzt die Beschreibungskomplexität von metalinearen CD Grammatiksystemen behandeln. Es stellen sich hier vor allen Dingen die Fragen danach, wie sich die Größe der Beschreibung einer bestimmten Sprachklasse ändert, wenn man von $(m+1)$-linearen CD Grammatiksystemen auf m-lineare CD Grammatiksysteme wechselt. Außerdem ist es nach dem im letzten Abschnitt bewiesenen Lemma über den Zusammenhang von metalinearen Sprachen und Sprachen, die durch metalineare CD Grammatiksysteme erzeugt werden können, interessant zu untersuchen, wie sich die Größe einer Beschreibung ändert, wenn man von einem m-linearen CD Grammatiksystem auf eine $(2m-1)$-lineare kontextfreie Grammatik wechselt.

Wir werden einige Ergebnisse aus dem Abschnitt über generative Mächtigkeit von metalinearen CD Grammatiksystemen nutzen, um die eben gestellten Fragen zu beantworten. Dazu werden für jede Turingmaschine M geeignete Sprachen basierend auf $INVALC(M)$ konstruiert und dann nichtrekursive Tradeoffs bewiesen.

Lemma 6.12 *Sei M eine Turingmaschine und $c \notin alph(INVALC(M))$. Sei weiterhin $f \in \{t, =k, \geq k \mid k \geq 2\}$, $m \geq 2$ und die Sprache L_1 definiert mit*

$$L_1 = (INVALC(M)c)^{2m-2}INVALC(M).$$

Es gilt

$$L_1 \in \mathcal{L}(CD\text{-}(m-1)LIN, f) \text{ genau dann, wenn } T(M) \text{ endlich ist.}$$

Beweis: Wenn $T(M)$ endlich ist, dann ist auch $VALC(M)$ endlich und daher das Komplement $INVALC(M)$ regulär. Nach den Abschlußeigenschaften regulärer Sprachen ist $(INVALC(M)c)^{2m-2}INVALC(M)$ ebenfalls regulär und daher auch linear kontextfrei. Alle linear kontextfreien Sprachen sind in der Klasse $\mathcal{L}(CD\text{-}(m-1)LIN, f)$ für $m \geq 2$.

Wenn $T(M)$ unendlich ist, ist nach Lemma 2.8 und Lemma 2.10 $INVALC(M)$ in $\mathcal{L}(LIN) \setminus \mathcal{L}(REG)$. Wenden wir jetzt Korollar 6.1 an, so können wir sehen, daß $L_1 \in \mathcal{L}(CD\text{-}mLIN, f) \setminus \mathcal{L}(CD\text{-}(m-1)LIN, f)$ ist. Insgesamt erhalten wir das Ergebnis. □

Lemma 6.13 *Sei M eine Turingmaschine und $c \notin alph(INVALC(M))$. Sei weiterhin $f \in \{t, =k, \geq k \mid k \geq 2\}$, $m \geq 2$ und die Sprache L_1 definiert wie in Lemma 6.12. Es gilt*

$$L_1 \in \mathcal{L}(CD\text{-}mLIN, f).$$

Beweis: Da $INVALC(M)$ nach Lemma 2.8 eine lineare Sprache ist, folgt der Satz direkt aus Korollar 6.1. □

Lemma 6.14 *Sei M eine Turingmaschine und $a, b, c \notin INVALC(M)$. Seien weiterhin $f \in \{t, =k, \geq k \mid k \geq 2\}$ und $m \geq 2$. Die Sprache L_2 ist definiert wie folgt:*

$$L_2 = \{a^n INVALC(M) c INVALC(M) b^n \mid n \geq 1\}.$$

Es gilt

$$L_2 \in \mathcal{L}(CD\text{-}mLIN, f).$$

Beweis: Sei $G = (N, T, P, S)$ eine linear kontextfreie Grammatik mit $L(G) = INVALC(M)$. Seien $A, A', B, B', S_\Gamma \notin N$. Wir betrachten jetzt das CD Grammatiksystem

$$\Gamma = (\{S_\Gamma, A, A', B, N^{(i)} \mid 1 \leq i \leq 2\}, T, P_1, \ldots, P_5, S_\Gamma)$$

mit

$$\begin{aligned}
P_1 &= \{C^{(1)} \to \alpha^{(1)}, \mid C \to \alpha \in P\} \cup \\
&\quad \{C^{(2)} \to \alpha^{(2)}, \mid C \to \alpha \in P\}, \\
P_2 &= \{S_\Gamma \to A'B, A' \to Ac\}, \\
P_3 &= \{A \to aA', B \to B'b\}, \\
P_4 &= \{A' \to A, B' \to B\}, \\
P_5 &= \{A' \to S^{(1)}, B' \to S^{(2)}\}.
\end{aligned}$$

Die Homomorphismen (1) und (2) sind dabei wie im zweiten Kapitel definiert.

Γ arbeitet im t-Ableitungsmodus. Die Ableitungen von Γ müssen mit P_2 starten und nach Anwendung beider Produktionen eine Satzform AcB erreichen. Danach muß die Komponente P_3 angewendet werden, die genau ein a und ein b erzeugt. Insgesamt erhalten wir $aA'cB'b$. Als nächstes können beliebig oft im Wechsel P_4 und P_3 angewendet werden. Die Komponente P_5 schließt dann die Erzeugung einer gleichen Anzahl von Symbolen a und b ab. Nach der Anwendung von P_5 haben wir eine Satzform $a^n S^{(1)} c S^{(2)} b^n$ und können nun nur noch die Komponente P_1 anwenden. P_1 enthält zwei Kopien von G, die von den Nichtterminalen her disjunkt sind. Nach den Regeln des t-Modus muß jetzt mit der Komponente P_1 an jedem der Nichtterminale $S^{(1)}$ und $S^{(2)}$ ein Wort aus $INVALC(M)$ erzeugt werden, bevor die Ableitung beendet ist.

Eine Ableitung in Γ sieht also wie folgt aus:

$$S \quad \overset{t}{\Longrightarrow}_{P_2} AcB \overset{t}{\Longrightarrow}_{P_3} aA'cB'b(\overset{t}{\Longrightarrow}_{P_4} \ldots \overset{t}{\Longrightarrow}_{P_3} \ldots)^{n-1}a^nA'cB'b^n$$
$$\overset{t}{\Longrightarrow}_{P_5} a^nS^{(1)}cS^{(2)}b^n \overset{t}{\Longrightarrow}_{P_1} a^nxcyb^n,$$

wobei $x, y \in \mathit{INVALC}(M)$ und $n \geq 1$. Da Γ im t-Modus keine anders geformten Wörter erzeugen kann, ist $L_2 = L(\Gamma, t)$.

Alle Produktionen außer der Startproduktion sind linear, und die Startproduktion enthält zwei Nichtterminale. Also ist Γ ein 2-lineares CD Grammatiksystem. Nach Satz 6.3 können wir auch für jeden anderen Ableitungsmodus $f \in \{t, =k \geq k \mid k \geq 2\}$ ein 2-lineares CD Grammatiksystem finden, das L_2 im Modus f erzeugt. Damit gilt $L_2 \in \mathcal{L}(\mathit{CD\text{-}mLIN}, f)$ für $f \in \{t, =k, \geq k \mid k \geq 2\}$ und $m \geq 2$. □

Lemma 6.15 *[Mal] Sei M eine Turingmaschine und seien a, b, c Symbole, die nicht in $alph(\mathit{INVALC}(M))$ sind. Sei weiterhin die Sprache L_2 definiert wie in Lemma 6.14. Es gilt für alle $m \geq 1$*

$L_2 \in \mathcal{L}(mLIN)$ *genau dann, wenn $T(M)$ endlich ist.*

Wir können jetzt die aus $\mathit{INVALC}(M)$ für jede Turingmaschine M konstruierten Sprachen benutzen und die im Kapitel 2 vorgestellte Technik von Hartmanis aus [Har79] nutzen, um nichtrekursive Tradeoffs zu beweisen.

Satz 6.9 *Sei $f \in \{t, =k, \geq k \mid k \geq 2\}$ und $m \geq 2$, dann gilt*

$$(\mathit{CD\text{-}}(m-1)\mathit{LIN}, f) \overset{nonrec}{\longleftarrow} (\mathit{CD\text{-}m\mathit{LIN}}, f).$$

Beweis: Wir können für jede Turingmaschine M eine Sprache L_1, die definiert ist wie in Lemma 6.12, effektiv konstruieren. Aus Lemma 6.13 wissen wir, daß $L_1 \in \mathcal{L}(\mathit{CD\text{-}m\mathit{LIN}}, f)$ ist.

Weiterhin ist nach Lemma 6.12 $L_1 \in \mathcal{L}(\mathit{CD\text{-}}(m-1)\mathit{LIN}, f)$ genau dann, wenn $T(M)$ endlich ist. Jetzt können wir Satz 2.7 anwenden und erhalten den nichtrekursiven Tradeoff. □

Satz 6.10 *Sei $f \in \{t, =k, \geq k \mid k \geq 2\}$ und $m \geq 2$, dann gilt*

$$(2m-1)\mathit{LIN} \overset{nonrec}{\longleftarrow} (\mathit{CD\text{-}m\mathit{LIN}}, f).$$

Beweis: Wir können für jede Turingmaschine M die in Lemma 6.14 definierte Sprache L_2 effektiv konstruieren. Nach Lemma 6.14 gilt $L_2 \in \mathcal{L}(\mathit{CD\text{-}m\mathit{LIN}}, f)$ für alle $m \geq 2$.

Außerdem ist in Lemma 6.15 gezeigt, daß $L_2 \in \mathcal{L}((2m-1)\mathit{LIN})$ genau dann gilt, wenn $T(M)$ endlich ist. Mit Satz 2.7 erhalten wir dann den nichtrekursiven Tradeoff. □

Im Beweis von Satz 7.4 wird gezeigt, daß ein nichtrekursiver Tradeoff zwischen metalinearen CD Grammatiksystemen und Grammatiksystemen von endlichem Index existiert. Dieser Tradeoff gilt dann natürlich auch zwischen metalinearen CD Grammatiksystemen und kontextfreien CD Grammatiksystemen. Wir erhalten also den folgenden Satz.

Satz 6.11 *Sei $f \in \{t, =k, \geq k \mid k \geq 2\}$, dann gilt*

$$(\textit{CD-METALIN}, f) \overset{nonrec}{\longleftarrow} (\textit{CD-CF}, f).$$

Insgesamt erhalten wir bei allen Inklusionen zwischen Sprachklassen, die im zweiten Abschnitt dieses Kapitels bewiesen wurden, nichtrekursive Tradeoffs. Sie werden in Abbildung 6.3 nochmal zusammengefaßt. Aus den Sprachen,

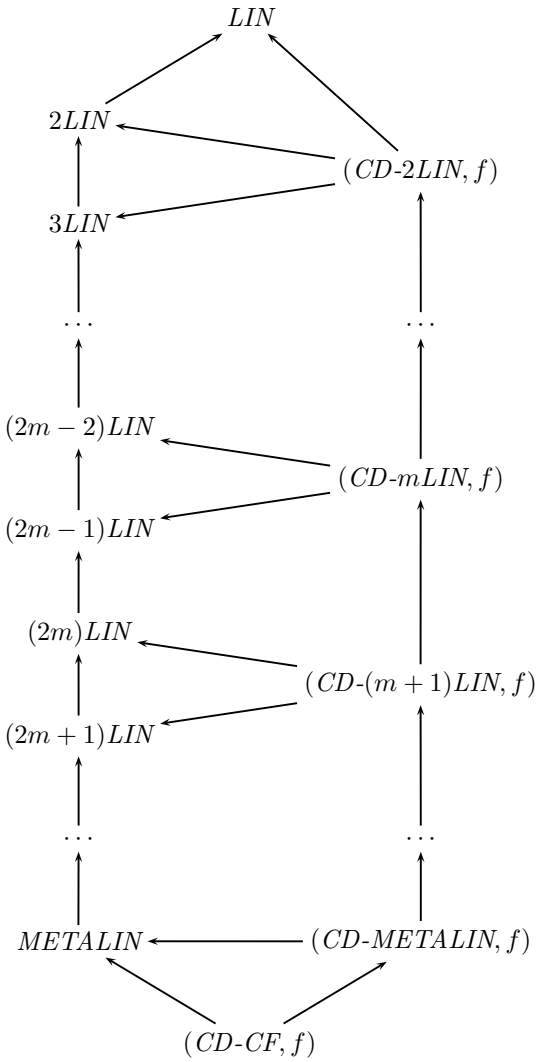

Abbildung 6.3: Nichtrekursive Tradeoffs bei metalinearen CD Grammatiksystemen und metalinearen kontextfreien Grammatiken

die wir für die Beweise der nichtrekursiven Tradeoffs benutzt haben, ergeben

sich auch Ergebnisse über die Entscheidbarkeit einiger Fragen im Zusammenhang mit metalinearen CD Grammatiksystemen.

Korollar 6.2 *Sei $f \in \{t, =k, \geq k \mid k \geq 2\}$, dann ist es unentscheidbar, ob*

1. *eine Sprache $L \in \mathcal{L}(CD\text{-}mLIN, f)$ auch in $\mathcal{L}(CD\text{-}(m-1)LIN, f)$ ist,*
2. *eine Sprache $L \in \mathcal{L}(CD\text{-}mLIN, f)$ auch in $\mathcal{L}((2m-2)LIN)$ ist,*
3. *eine Sprache $L \in \mathcal{L}(CD\text{-}CF, f)$ auch in $\mathcal{L}(CD\text{-}METALIN, f)$ ist.*

Beweis: 1. Aus Lemma 6.12 und Lemma 6.13 kennen wir eine Sprache, die man für jede Turingmaschine M konstruieren kann. Diese Sprache ist in der Klasse $\mathcal{L}(CD\text{-}mLIN, f)$ und genau dann in $\mathcal{L}(CD\text{-}(m-1)LIN, f)$, wenn $T(M)$ endlich ist. Könnte man also entscheiden, ob eine Sprache aus $\mathcal{L}(CD\text{-}mLIN, f)$ in $\mathcal{L}(CD\text{-}(m-1)LIN, f)$ ist, so könnte man entscheiden, ob $T(M)$ endlich ist. Das ist ein Widerspruch zu Satz 2.5.

2. Aus Lemma 6.14 und Lemma 6.15 kennen wir eine Sprache, die man für jede Turingmaschine M konstruieren kann. Diese Sprache ist in $\mathcal{L}(CD\text{-}mLIN, f)$ und genau dann in $\mathcal{L}((2m-1)LIN)$, wenn $T(M)$ endlich ist. Das Ergebnis erhält man mit den gleichen Argumenten wie in 1.

3. Aus Lemma 7.4 kennen wir eine Sprache, die man für jede Turingmaschine M konstruieren kann. Diese Sprache ist in $\mathcal{L}(CD\text{-}CF, f)$ und genau dann in der Klasse $\mathcal{L}(CD\text{-}METALIN, f)$, wenn $T(M)$ endlich ist. Das Ergebnis erhält man mit den gleichen Argumenten wie in 1. □

6.4 Zusammenfassung

Nach kontextfreien und hybriden CD Grammatiksystemen haben wir in diesem Kapitel metalineare CD Grammatiksysteme genau untersucht. Im Abschnitt über die generative Mächtigkeit haben wir gesehen, daß bei m-linearen CD Grammatiksystemen alle starken Ableitungsmodi $f \in \{t, =k, \geq k \mid k \geq 2\}$ die gleiche Sprachklasse erzeugen. Diese Klasse fällt auch mit der Klasse der von m-linearen ET0L Systemen und m-linearen Matrixgrammatiken generierten Sprachen zusammen. Weiterhin existiert eine unendliche Hierarchie von Sprachklassen über der Breite m von metalinearen CD Grammatiksystemen. Außerdem konnten wir für die betrachteten Sprachklassen Pumpinglemmata zeigen und durch die Abschlußeigenschaften sehen, daß $\mathcal{L}(CD\text{-}mLIN, f)$ eine volle semi-AFL ist. Ein weiteres Lemma über die Fähigkeit von m-linearen CD Grammatiksystemen, $(2m-1)$-lineare kontextfreie Sprachen zu erzeugen, wurde am Ende des Abschnitts über die generative Mächtigkeit bewiesen.

Im Abschnitt über die Beschreibungskomplexität haben wir untersucht, wie sich die Größe eines metalinearen CD Grammatiksystems ändert, wenn man eine Sprachklasse mit m-linearen bzw. $(m-1)$-linearen CD Grammatiksystemen beschreibt. Als Ergebnis erhielten wir einen nichtrekursiven Tradeoff. Einen weiteren nichtrekursiven Tradeoff konnten wir zeigen, wenn man eine Sprachklasse mit m-linearen CD Grammatiksystemen bzw. $(2m-1)$-linearen

kontextfreien Grammatiken beschreibt. Zuletzt konnten wir auch einen nichtrekursiven Tradeoff zwischen kontextfreien CD Grammatiksystemen und metalinearen CD Grammatiksystemen nachweisen. Alle die Beschreibungskomplexität von metalinearen CD Grammatiksystemen betreffenden Ergebnisse sind in Abbildung 6.3 zusammengefaßt. Darüber hinaus wurde gezeigt, daß die Frage, ob eine Sprache aus der stärkeren Sprachklasse auch in der schwächeren Sprachklasse enthalten ist, bei den Klassen, zwischen denen wir nichtrekursive Tradeoffs erhalten haben, nicht entscheidbar ist.

Im Gegensatz zu kontextfreien und hybriden CD Grammatiksystemen konnten wir für die Klasse der metalinearen CD Grammatiksysteme fast alle Fragen der generativen Mächtigkeit beantworten und haben auch ausführliche Ergebnisse in der Beschreibungskomplexität erhalten. Wir wollen im nächsten Kapitel versuchen, auch für größere Klassen von CD Grammatiksystemen ähnlich gute Ergebnisse zu erzielen.

Kapitel 7

CD Grammatiksysteme von endlichem Index

Im letzten Kapitel haben wir uns ausführlich mit metalinearen CD Grammatiksystemen beschäftigt. Durch die relativ starke Einschränkung in der Form der Produktionen ist eine Klasse von Grammatiksystemen entstanden, für die wir sowohl auf dem Gebiet der generativen Mächtigkeit als auch auf dem Gebiet der Beschreibungskomplexität gute Ergebnisse erzielen konnten.

Als nächstes stellt sich die Frage, ob wir auch für Klassen von CD Grammatiksystemen mit einer größeren generativen Mächtigkeit ähnlich gute Aussagen machen können. Wir wenden uns deshalb einer anderen Klasse von CD Grammatiksystemen zu, die mächtiger, aber trotzdem von der Handhabung her ähnlich einfach wie die Klasse der metalinearen CD Grammatiksysteme ist.

Wenn wir kontextfreie CD Grammatiksysteme so einschränken, daß es mindestens eine Ableitung für jedes in der erzeugten Sprache enthaltene Wort w geben muß, deren Satzformen höchstens m Nichtterminale enthalten, erhalten wir die Klasse der CD Grammatiksysteme vom Index m, die wir in diesem Kapitel untersuchen. Nach den nötigen Definitionen und einem Beispiel fassen wir die Ergebnisse über die generative Mächtigkeit zusammen. Im wichtigsten Teil dieses Kapitels werden dann mehrere Resultate aus der Beschreibungskomplexität bewiesen.

Die Ergebnisse über die generative Mächtigkeit sind hauptsächlich aus [DP90] und [CVDKP94]. Einige Ergänzungen finden sich in [Sun05a]. Außerdem werden dort alle Resultate aus der Beschreibungskomplexität von CD Grammatiksystemen von endlichem Index bewiesen.

7.1 Definitionen und Beispiel

Definition 7.1 *Sei $\Gamma = (N, T, P_1, \ldots P_n, S)$ ein CD Grammatiksystem und $f \in \{*, t, =k, \geq k, \leq k \mid k \geq 1\}$ ein Ableitungsmodus. Für die Ableitung*

$$D : S = \alpha_0 \overbrace{\Rightarrow_{P_{i_1}} \alpha_1 \Rightarrow P_{i_1}\alpha_2 \Rightarrow_{P_{i_1}}}^{f} \ldots \overbrace{\Rightarrow_{P_{i_p}} \alpha_{m-2} \Rightarrow_{P_{i_p}} \alpha_{m-1} \Rightarrow_{P_{i_p}}}^{f} \alpha_m,$$

mit $m, p \geq 0$ *und* $i_1, \ldots, i_p \in \{1, \ldots, n\}$ *in* Γ *nach Ableitungsmodus* f *ist*

$$Ind(D, f) = max\{|\alpha_j|_N \mid 1 \leq j \leq m\}.$$

Sei $D(\alpha, \Gamma, f)$ *die Menge aller Ableitungen von* α *in* Γ *mit Ableitungsmodus* f*. Für eine Satzform* α *von* Γ*, die im Ableitungsmodus* f *erreicht wurde, definieren wir den Index mit*

$$Ind(\alpha, f) = min\{Ind(D, f) \mid D \in D(\alpha, \Gamma, f)\}.$$

Der Index von Γ *im Ableitungsmodus* f *ist*

$$Ind(\Gamma, f) = sup\{Ind(w, f) \mid w \in L(\Gamma, f)\}.$$

Die Klasse der CD Grammatiksysteme, die im Ableitungsmodus f einen endlichem Index haben, wird mit $(\textit{CD-FIN}, f)$ bezeichnet, die Klasse der CD Grammatiksysteme mit Index m im Ableitungsmodus f mit $(\textit{CD-mFIN}, f)$. Die entsprechenden Sprachklassen nennen wir $\mathcal{L}(\textit{CD-FIN}, f)$ und $\mathcal{L}(\textit{CD-mFIN}, f)$.

Beispiel 7.1 Sei $L = \{a^n b^n \mid n \geq 1\}^*$. Das CD Grammatiksystem

$$\begin{aligned}
\Gamma_1 &= (\{S, A, B\}, \{a, b\}, P_1, P_2, S), \\
P_1 &= \{S \to AB, S \to S, S \to \varepsilon\}, \\
P_2 &= \{A \to aAb, A \to A, A \to ab, B \to S\}
\end{aligned}$$

erzeugt L im t-Modus. Eine Ableitung in Γ_1 sieht beispielsweise wie folgt aus:

$$\begin{aligned}
S \quad &\overset{t}{\Longrightarrow}_{P_1} AB \overbrace{(\Longrightarrow_{P_2} \ldots)^{n_1} a^{n_1} b^{n_1} B \Longrightarrow_{P_2}}^{t} a^{n_1} b^{n_1} S \overset{t}{\Longrightarrow}_{P_1} a^{n_1} b^{n_1} AB \\
&\overbrace{(\Longrightarrow_{P_2} \ldots)^{n_2} a^{n_1} b^{n_1} a^{n_2} b^{n_2} B \Longrightarrow_{P_2}}^{t} a^{n_1} b^{n_1} a^{n_2} b^{n_2} S \overset{t}{\Longrightarrow}_{P_1} a^{n_1} b^{n_1} a^{n_2} b^{n_2}.
\end{aligned}$$

Zuerst wird S mit P_1 durch AB ersetzt. Danach wird die Komponente P_2 benutzt. Im t-Modus muß sie B durch S ersetzen und A durch $a^{n_1} b^{n_1}$, wobei $n_1 \geq 1$. Im nächsten Schritt wird wieder P_1 aktiv und ersetzt S durch AB. In unserer Beispielableitung wird nach dem nochmaligen Gebrauch von P_2 die Ableitung beendet. Es ist allerdings möglich, auf ähnliche Weise beliebig viele konkatenierte Paare $a^n b^n$ mit $n \geq 1$ zu erzeugen. Man kann sehen, daß insgesamt in allen Satzformen im t-Ableitungsmodus nur zwei Nichtterminale vorhanden sind. Daher ist $Ind(\Gamma, t) = 2$ und $L(\Gamma, t) = L$.

Das gleiche CD Grammatiksystem kann auch im $(= 2)$-Modus benutzt werden und erzeugt die gleiche Sprache wie im t-Modus. Hier können zwar auch Satzformen mit mehr als zwei Nichtterminalen vorkommen. Allerdings ist der Index über die Ableitung mit der minimalen Anzahl von Nichtterminalen definiert. Es gilt deshalb auch $Ind(\Gamma, =2) = 2$.

7.2 Generative Mächtigkeit

In diesem Abschnitt werden wir die schon bekannten Ergebnisse über die generative Mächtigkeit von CD Grammatiksystemen von endlichem Index aus [DP90] und [CVDKP94] zusammenfassen. Diese Ergebnisse werden ergänzt, wenn in den ursprünglichen Konstruktionen der Index der Grammatiksysteme nicht beibehalten wurde.

Wie bei metalinearen CD Grammatiksysteme gilt auch hier, daß die Ableitungsmodi t, $(= k)$ und $(\geq k)$ für $k \geq 2$ äquivalent sind. Außerdem fällt die Klasse der erzeugten Sprachen mit der Klasse zusammen, die von ET0L-Systemen und Matrixgrammatiken vom Index m erzeugt wird. Es können also die Ergebnisse über Matrixgrammatiken von endlichem Index aus [DP89] und *ET0L*-Systeme von endlichem Index aus [RV78b] und [RV78a] übertragen werden.

Lemma 7.1 *Sei $f \in \{*, = 1, \geq 1, \leq k \mid k \geq 1\}$. Es gilt*

$$\mathcal{L}(\textit{mFIN}) = \mathcal{L}(\textit{CD-mFIN}, f)$$

für $m \geq 1$.

Beweis: Wenn wir die Beweise für Satz 4.1 auf kontextfreie Grammatiken von endlichem Index bzw. CD Grammatiksysteme von endlichem Index anwenden, wird der Index nicht verändert. Wir erhalten die obige Aussage. □

Lemma 7.2 *Sei $f \in \{*, t, \leq k, = k, \geq k \mid k \geq 1\}$. Es gilt*

$$\mathcal{L}(\textit{LIN}) = \mathcal{L}(\textit{CD-1FIN}, f).$$

Beweis: Sei L eine Sprache in $\mathcal{L}(\textit{LIN})$ und $G = (N, T, P, S)$ eine linear kontextfreie Grammatik mit $L(G) = L$, dann gibt es ein CD Grammatiksystem $\Gamma = (N, T, P', S)$ mit nur einer Komponente $P' = P \cup \{A \to A \mid A \in N\}$, die aus den Produktionen der linear kontextfreien Grammatik G und den Kettenproduktionen $A \to A$ besteht. Γ erzeugt L in jedem beliebigen Ableitungsmodus f. Außerdem besteht P' nur aus linearen Produktionen und deshalb ist $Ind(\Gamma, f) = 1$.

Sei jetzt $\Gamma \in \mathcal{L}(\textit{CD-1FIN}, f)$ und $L = L(\Gamma, f)$. Zuerst können wir das CD Grammatiksystem $\Gamma = (N, T, P_1, \ldots, P_n, S)$ in ein lineares CD Grammatiksystem

$$\Gamma' = (N \cup \{F\}, T, P'_1, \ldots, P'_n, S)$$

umwandeln. Dabei ist $F \notin N \cup T$ ein neues Nichtterminal und für $1 \leq i \leq n$ die Komponente

$$\begin{aligned} P'_i \;=\; & \{A \to \alpha \mid A \to \alpha \in P_i, A \in N, \alpha \in T^* N^{\leq 1} T^*\} \cup \\ & \{B \to F \mid B \to \beta \in P_i, B \in N, \beta \in (N \cup T)^* N (N \cup T)^* N (N \cup T)^*\}. \end{aligned}$$

Alle Ableitungen von Γ', die ein Wort $w \in T^*$ erzeugen, bestehen nur aus den linearen Produktionen von Γ. Es gilt also $L(\Gamma', f) \subseteq L(\Gamma, f)$ Da laut Definition

des Index eines CD Grammatiksystems jedes Wort in L eine Ableitung in Γ hat, in deren Satzformen höchstens ein Nichtterminal vorkommt, können alle Wörter $w \in L$ auch von Γ' erzeugt werden. Es ist also $L(\Gamma, f) \subseteq L(\Gamma', f)$.

Satz 6.1 sagt aus, daß jedes lineare CD Grammatiksystem eine linear kontextfreie Sprache erzeugt. □

Lemma 7.3 *Sei $d \in \{= k, \geq k \mid k \geq 2\}$. Es gilt*

$$\begin{aligned} \mathcal{L}(\textit{CD-mFIN}, t) &= \mathcal{L}(\textit{ET0L-mFIN}) \\ &= \mathcal{L}(\textit{MAT-mFIN}) = \mathcal{L}(\textit{CD-mFIN}, d) \end{aligned}$$

für alle $m \geq 1$.

Beweis: Wir können die Konstruktionen in den Beweisen für Satz 4.2 und Satz 4.3 auf den Fall des Index m übertragen, ohne den Index zu ändern, und haben damit $\mathcal{L}(\textit{CD-mFIN}, t) = \mathcal{L}(\textit{ET0L-mFIN})$ und auch $\mathcal{L}(\textit{CD-mFIN}, d) \subseteq \mathcal{L}(\textit{MAT-mFIN})$. In [CVDKP94] wird gezeigt, daß $\mathcal{L}(\textit{MAT-FIN}) \subseteq \mathcal{L}(\textit{CD-FIN}, d)$ gilt. Da auch bei dieser Konstruktion der Index erhalten wird, ergibt sich die Inklusion $\mathcal{L}(\textit{MAT-mFIN}) \subseteq \mathcal{L}(\textit{CD-mFIN}, d)$. Darüber hinaus wird die Äquivalenz $\mathcal{L}(\textit{ET0L-mFIN}) = \mathcal{L}(\textit{MAT-mFIN})$ in [RV78b] bewiesen. Insgesamt erhalten wir den obigen Satz. □

Lemma 7.4 *Sei $f \in \{t, = k, \geq k \mid k \geq 2\}$ und $m \geq 2$, dann ist*

$$\mathcal{L}(\textit{CD-mLIN}, f) \subset \mathcal{L}(\textit{CD-mFIN}, f).$$

Beweis: Die Inklusion gilt, da jedes m-lineare CD Grammatiksystem den Index m hat. Aus Beispiel 7.1 wissen wir, daß die Sprache $L = \{a^n b^n \mid n \geq 1\}^* \in \mathcal{L}(\textit{CD-mFIN}, t)$ für $m \geq 2$. Wegen Lemma 7.3 ist $L \in \mathcal{L}(\textit{CD-mFIN}, f)$. Es wird aber in Satz 6.4 gezeigt, daß $L \notin \mathcal{L}(\textit{CD-mLIN}, f)$ für $m \geq 2$. Zusammen ergibt sich die echte Inklusion. □

Das folgende Pumpinglemma wird in [DP89] für Matrixgrammatiken vom Index m bewiesen und gilt daher auch für CD Grammatiksysteme vom Index m.

Lemma 7.5 *Für jede unendliche Sprache $L \in \mathcal{L}(\textit{CD-mFIN}, f)$ gibt es eine Konstante $p \geq 0$, so daß alle $z \in L$ mit $|z| > p$ geschrieben werden können als*

$$z = u_1 v_1 w_1 x_1 u_2 v_2 w_2 x_2 \ldots u_n v_n w_n x_n u_{n+1}$$

mit $n \leq m$, $|v_1 x_1 v_2 x_2 \ldots v_k x_k| > 0$ und

$$u_1 v_1^i w_1 x_1^i u_2 v_2^i w_2 x_2^i \ldots u_n v_n^i w_n x_n^i u_{n+1} \in L$$

für alle $i \geq 0$.

Mit Hilfe des das Pumpinglemmas kann man zeigen, daß durch den Index eine unendliche Hierarchie von Sprachklassen entsteht.

Lemma 7.6 *[CVDKP94] Sei $f \in \{t, = k, \geq k \mid k \geq 2\}$, dann ist*

$$\mathcal{L}(\textit{CD-mFIN}, f) \subset \mathcal{L}(\textit{CD-(m+1)FIN}, f)$$

für alle $m \geq 1$.

Einige weitere Ergebnisse, die für Matrixgrammatiken von endlichem Index und damit auch für CD Grammatiksysteme von endlichem Index gelten, werden wir im folgenden auch benutzen.

Lemma 7.7 *[DP89] Es gibt einen Algorithmus, der für ein CD Grammatiksystem $\Gamma = (N, T, P_1, \ldots, P_n, S) \in$ CD-mFIN und einen String $w \in T^*$ bestimmt, ob $w \in L(\Gamma, f)$.*

Unter anderem sind folgende Abschlußeigenschaften bekannt.

Satz 7.1 *[DP89] Seien $f \in \{t, = k, \geq k \mid k \geq 2\}$ und $m \geq 1$, dann ist die Sprachklasse $\mathcal{L}(\textit{CD-}mFIN, f)$ unter den Operationen*

1. *Vereinigung*
2. *Homomorphismus*
3. *Schnitt mit regulären Mengen*
4. *inversem Homomorphismus*

abgeschlossen und unter den Operationen

1. *Konkatenation*
2. *Kleenescher Hülle*
3. *Schnitt*
4. *Komplement*

nicht abgeschlossen.

Insgesamt ergibt sich für die besprochenen Sprachklassen, die von CD Grammatiksystemen von endlichem Index und von metalinearen CD Grammatiksystemen erzeugt werden, die in Abbildung 7.1 dargestellte Hierarchie.

In [DP89] wird gezeigt, daß es auch kontextfreie Sprachen gibt, die nicht in $\mathcal{L}(\textit{CD-mFIN}, f)$ für $f \in \{t, = k, \geq k \mid k \geq 2\}$ und $m \geq 1$ enthalten sind. Diese Sprachen werden in [ER77] genauer untersucht. Es handelt sich dabei um die „binary bracketed“ Sprachen, die unter anderem die Dyck-Sprache enthalten.

7.3 Beschreibungskomplexität

Wenn wir jetzt CD Grammatiksysteme von endliche Index betrachten, stellen sich hinsichtlich der Beschreibungskomplexität mehrere Fragen. Es gibt Sprachen, die wir mit einem CD Grammatiksystem vom Index m und auch mit einem CD Grammatiksystem vom Index $m-1$ darstellen können. Wie verändert

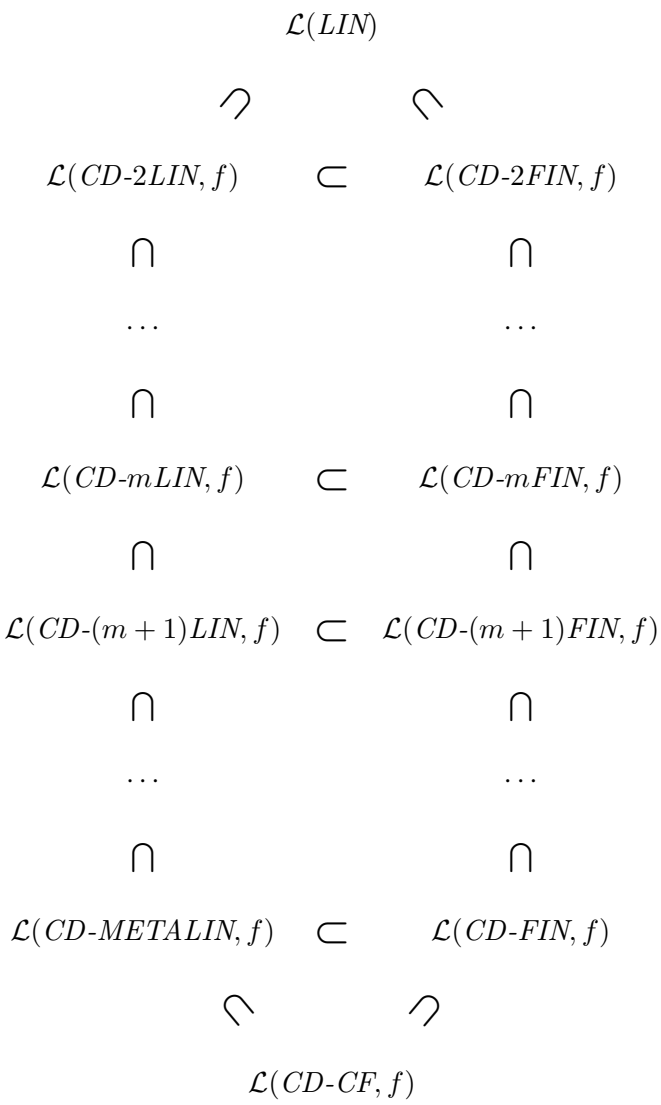

Abbildung 7.1: Die generative Mächtigkeit von CD Grammatiksystemen von endlichem Index

sich die Größe der Beschreibung, wenn wir vom stärkeren zum schwächeren Beschreibungssystem wechseln? Weiterhin gibt es Sprachen, die mit einem CD Grammatiksystem vom Index m und einem m-linearen CD Grammatiksystem dargestellt werden können. Wie verändert sich die Größe der Beschreibung, wenn wir vom stärkeren zum schwächeren Beschreibungssystem wechseln? Mit diesen Fragen werden wir uns in diesem Abschnitt beschäftigen.

Zuerst werden zwischen CD Grammatiksystemen vom Index m und vom Index $m-1$ nichtrekursive Tradeoffs bewiesen. Außerdem erhalten wir auch für m-lineare CD Grammatiksysteme und CD Grammatiksysteme vom Index m nichtrekursive Tradeoffs. Dazu müssen einige Eigenschaften von Sprachen gezeigt werden, die in geeigneter Weise aus $\mathit{INVALC}(M)$ für eine beliebige Turingmaschine M konstruiert werden.

Lemma 7.8 *Für jede Turingmaschine M und ein $c \notin alph(\mathit{INVALC}(M))$ sei*

$$L_1 = (\mathit{INVALC}(M)c)^{2m}\mathit{INVALC}(M).$$

Für alle $m \geq 2$ und $f \in \{t, = k, \geq k \mid k \geq 2\}$ gilt

$$L_1 \in \mathcal{L}(\mathit{CD\text{-}mFIN}, f).$$

Beweis: Sei $G = (N, T, P, S)$ eine linear kontextfreie Grammatik und gelte $L(G) = \mathit{INVALC}(M)$. Weiterhin seien $S', B^{[2]}, \ldots, B^{[2m]} \notin N$. Wir werden jetzt das CD Grammatiksystem Γ_1 vom Index 2 konstruieren, das L_1 im t-Modus erzeugt. Dabei sind die Homomorphismen $(i) : N \cup T \to N^{(i)} \cup T$ mit $1 \leq i \leq 2m+1$ definiert wie in Kapitel 2 beschrieben.

Γ_1 wird wie folgt konstruiert:

$$\begin{aligned}\Gamma_1 \;=\; & (N^{(1)} \cup \ldots \cup N^{(2m+1)} \cup \{B^{[2]}, \ldots, B^{[2m]}, S'\},\\ & T, P_1, \ldots, P_{4m}, S'),\end{aligned}$$

und die Komponenten setzen sich wie folgt zusammen:

$$\begin{aligned}
P_1 &= \{S' \to S^{(1)}cB^{[2]}\},\\
P_2 &= \{A^{(1)} \to \alpha^{(1)} \mid A \to \alpha \in P\},\\
P_3 &= \{B^{[2]} \to S^{(2)}cB^{[3]}\},\\
P_4 &= \{A^{(2)} \to \alpha^{(2)} \mid A \to \alpha \in P\},\\
&\ldots\\
P_{4m-1} &= \{B^{[2m]} \to S^{(2m)}cS^{(2m+1)}\},\\
P_{4m} &= \{A^{(2m)} \to \alpha^{(2m)} \mid A \to \alpha \in P\},\\
P_{4m+1} &= \{A^{(2m+1)} \to \alpha^{(2m+1)} \mid A \to \alpha \in P\}.
\end{aligned}$$

Im t-Modus benutzt Γ_1 zuerst P_1 um die Nichtterminale $S^{(1)}$ und $B^{[2]}$ zu erzeugen. Danach wird P_2 zur aktiven Komponente. Sie muß benutzt werden, bis $S^{(1)}$ durch einen Terminalstring ersetzt ist. Da die Produktionen $A \to \alpha$ aus

der linearen Grammatik G stammen, sind bei jedem dieser Ableitungsschritte höchstens zwei Nichtterminale in der Satzform vorhanden. Danach wird P_3 benutzt. P_3 und P_4 arbeiten genauso wie P_1 und P_2 und so fort. Die letzten beiden Teilstrings im entstehenden Wort werden von P_{4m} und P_{4m+1} erzeugt. Insgesamt sieht eine Ableitung in Γ_2 wie folgt aus:

$$\begin{aligned} S' \quad & \overset{t}{\Longrightarrow}_{P_1} S^{(1)}cB^{[2]} \overset{t}{\Longrightarrow}_{P_2} w_1cB^{[2]} \overset{t}{\Longrightarrow}_{P_3} w_1cS^{(2)}cB^{[3]} \overset{t}{\Longrightarrow}_{P_4} w_1cw_2cB^{[3]} \\ & \overset{t}{\Longrightarrow}_{P_5} \ldots \overset{t}{\Longrightarrow}_{P_{4m-2}} w_1cw_2c\ldots cw_{2m-1}cB^{[2m]} \\ & \overset{t}{\Longrightarrow}_{P_{4m-1}} w_1cw_2c\ldots cw_{2m-1}cS^{(2m)}cS^{(2m+1)} \\ & \overset{t}{\Longrightarrow}_{P_{4m}} w_1cw_2c\ldots cw_{2m-1}cw_{2m}cS^{(2m+1)} \\ & \overset{t}{\Longrightarrow}_{P_{4m+1}} w_1cw_2c\ldots cw_{2m-1}cw_{2m}cw_{2m+1}, \end{aligned}$$

wobei $w_1, w_2, \ldots, w_{2m+1} \in L(G) = \mathit{INVALC}(M)$.

Insgesamt sind in den Satzformen der Ableitung immer höchstens zwei Nichtterminale vorhanden. Es gilt also nicht nur $L(\Gamma_1, t) = L_1$, sondern es ist auch $\mathit{Ind}(\Gamma_1, t) = 2$. Aus Lemma 7.3 wissen wir, daß es auch für die übrigen Ableitungsmodi ein CD Grammatiksystem vom Index m gibt, das L_1 erzeugt. □

Mit dem eben gezeigten Lemma und einem Ergebnis aus dem vorigen Kapitel werden wir später die nichtrekursiven Tradeoffs zwischen CD Grammatiksystemen vom Index m und m-linearen CD Grammatiksystemen zeigen.

Die folgenden Lemmata beinhalten wieder die Eigenschaften einer für jede Turingmaschine M aus $\mathit{INVALC}(M)$ geeignet konstruierten Sprache.

Lemma 7.9 *Sei M eine Turingmaschine und seien $h' : alph(\mathit{INVALC}(M)) \rightarrow alph(\mathit{INVALC}(M))'$ und $h'' : alph(\mathit{INVALC}(M)) \rightarrow alph(\mathit{INVALC}(M))''$ zwei Homomorphismen mit $h'(c) = c'$ und $h''(c) = c''$ für $c \in$ alph(INVALC(M)). Wenn $T(M)$ unendlich ist, dann gilt*

$$L' = \{yz \mid y \in h'(\mathit{INVALC}(M), z \in h''(\mathit{INVALC}(M))\} \in \mathcal{L}(2LIN) \setminus \mathcal{L}(LIN).$$

Beweis: Wenn $T(M)$ unendlich ist, dann ist nach den Lemmata 2.8 und 2.10 $\mathit{INVALC}(M) \in \mathcal{L}(LIN) \setminus \mathcal{L}(REG)$. Durch die Abschlußeigenschaften von linearen Sprachen sind damit auch $h'(\mathit{INVALC}(M))$ und $h''(\mathit{INVALC}(M))$ in der Klasse $\mathcal{L}(LIN) \setminus \mathcal{L}(REG)$. Eine 2-lineare Grammatik für L' läßt sich leicht aus Grammatiken für $\mathit{INVALC}(M)$ konstruieren.

Sei jetzt $c \notin alph(\mathit{INVALC}(M))$. Aus Lemma 2.6 wissen wir, daß $L'' = \{ycz \mid y \in h'(\mathit{INVALC}(M)), z \in h''(\mathit{INVALC}(M))\}$ in $\mathcal{L}(2LIN) \setminus \mathcal{L}(LIN)$ ist. Man kann aber eine gsm-Abbildung σ konstruieren, für die $\sigma(L') = L''$ gilt. Dabei liest σ erst alle Symbole aus $h'(\mathit{INVALC}(M))$ und gibt sie unverändert aus. Sobald das erste Symbol aus $h''(\mathit{INVALC}(M))$ gelesen wird, gibt σ ein c gefolgt von dem gelesenen Symbol aus. Danach werden alle Symbole aus $h''(\mathit{INVALC}(M))$ gelesen und unverändert ausgegeben. Mit der endlichen Kontrolle stellt σ fest, wann das erste Symbol aus $h''(\mathit{INVALC}(M))$ gelesen wird. Wenn jetzt L' eine lineare Sprache wäre, dann wäre auch L'' linear, da lineare Sprachen nach

Lemma 2.9 unter gsm-Abbildungen abgeschlossen sind. Das widerspricht den vorherigen Bemerkungen, und somit ist $L' \in \mathcal{L}(2LIN) \setminus \mathcal{L}(LIN)$. □

Lemma 7.10 *Sei M eine Turingmaschine und a und b zwei Symbole mit $a, b \notin alph(INVALC(M))$. Seien dann h' und h'' zwei Homomorphismen mit $h'(c) = c'$ und $h''(c) = c''$ für $c \in alph(INVALC(M))$. Die Sprache L_2 ist wie folgt definiert:*

$$\begin{aligned} L_2 &= \{(a^n x b^n)^{m-1} a^n x^R xyz b^n \mid n \geq 1, \\ &\quad x \in INVALC(M), y \in h'(INVALC(M)), z \in h''(INVALC(M))\}. \end{aligned}$$

Für jeden Ableitungsmodus $f \in \{t, = k, \geq k \mid k \geq 2\}$ und $m \geq 2$ gilt

$L_2 \in \mathcal{L}(CD\text{-}mFIN, f)$ *genau dann, wenn $T(M)$ endlich ist.*

Beweis: Wenn $T(M)$ endlich ist, dann ist nach Lemma 2.10 $INVALC(M)$ regulär. Es gibt also eine linkslineare Grammatik

$$\begin{aligned} G_R &= (N_R, T, P_R, S_R) \text{ mit } L(G_R) = INVALC(M), \\ &\quad T = alph(INVALC(M)). \end{aligned}$$

Wir können dann für

$$\begin{aligned} L_2' &= \{x^R xyz \mid x \in INVALC(M), y \in h'(INVALC(M)), \\ &\quad z \in h''(INVALC(M))\} \end{aligned}$$

aus G_R eine lineare Grammatik

$$G_3 = (N_3, T_3, P_3, S_R^{(2)}) \text{ mit } L(G_3) = L_2'$$

konstruieren. Dabei ist $T_3 = T \cup h'(T) \cup h''(T)$ und $N_3 = N_R \cup N_R^{(1)} \cup N_R^{(2)}$ Die Produktionsmenge P_3 setzt sich wie folgt zusammen:

$$\begin{aligned} P_3 = {} & \{A^{(2)} \to B^{(2)} h''(u) \mid A, B \in N_R,\ u \in T,\ A \to Bu \in P_R\} \cup & (7.1) \\ & \{A^{(2)} \to S_R^{(1)} h''(u) \mid A \in N_R,\ u \in T,\ A \to u \in P_R\} \cup & (7.2) \\ & \{A^{(1)} \to B^{(1)} h'(u) \mid A, B \in N_R,\ u \in T,\ A \to Bu \in P_R\} \cup & (7.3) \\ & \{A^{(1)} \to S_R h'(u) \mid A \in N_R,\ u \in T,\ A \to u \in P_R\} \cup & (7.4) \\ & \{A \to u^R B u \mid A, B \in N_R,\ u \in T,\ A \to Bu \in P_R\} \cup & (7.5) \\ & \{A \to u^R u \mid A \in N_R,\ u \in T,\ A \to u \in P_R\} & (7.6) \end{aligned}$$

Eine Ableitung in G_3 beginnt mit $S_R^{(2)}$. Mit Produktionen aus (7.1) und einer abschließenden Produktion aus (7.2) wird der String $z \in h''(INVALC(M))$ erzeugt. Da die verwendeten Produktionen sich nur durch die Markierung $''$ an den Terminalen und die Markierung (2) an den Nichtterminalen von denen aus G_R unterscheiden, sind sie linkslinear. Es ist also möglich, bei der

abschließenden Produktion, die auf der rechten Seite nur Terminale hat, das Startsymbol für die Erzeugung des Strings $y \in h'(\mathit{INVALC}(M))$ anzuhängen. Die dazu benötigten Produktionen finden sich in (7.3) und die abschließende Produktion für y in (7.4). Auch diese Produktionen sind linkslinear, und es ist möglich, an die abschließende Produktion das Startsymbol für die Erzeugung von $x^R x, x \in \mathit{INVALC}(M)$ anzuhängen. Bei der Erzeugung von $x^R x$ werden wieder die Produktionen aus P_R verwendet. Die in diesen Produktionen vorhandenen Nichtterminale sind aber gespiegelt nochmal an das linke Ende der rechten Seite der jeweiligen Produktion angefügt. Auf diese Weise wird mit (7.5) und (7.6) nicht x, sondern $x^R x$ erzeugt. Insgesamt haben alle Ableitungen die folgende Form:

$$S_R^{(2)} \Rightarrow^* S_R^{(1)} z \Rightarrow^* S_R yz \Rightarrow^* x^R xyz.$$

Durch die Markierungen der Nichtterminale können keine Produktionen von (7.1) und (7.2) mit solchen aus (7.3) und (7.4) oder (7.5) und (7.6) vermischt werden. Außerdem besteht die erzeugte Grammatik nur aus linearen Produktionen. Die Sprache L_2' ist also linear.

Wir können jetzt aus der linkslinearen Grammatik G_R ein CD Grammatiksystem Γ_2, das L_2 erzeugt, konstruieren. Ein Teil dieses CD Grammatiksystems arbeitet dabei wie G_3. Es sind $S, C, C', D, D', E \notin N_R$ für $1 \leq i \leq p$, und es gilt $P_R = \{p_1, \ldots, p_p\}$. Weiterhin ist

$$\begin{aligned}
\Gamma_2 &= \{\{S', C, C', D, D', E\} \cup N_R \cup N_R' \cup N_R^{(1)} \cup N_R^{(2)} \cup N_R^{(3)} \cup N_R^{(4)}, \{a, b\}, \\
&\quad P_1, P_2, P_3, P_4, P_5, P_6, Q_1, \ldots, Q_p, S\} \\
P_1 &= \{S \to C^{m-1} D\}, \\
P_2 &= \{C \to aC'b, D \to aD'b\}, \\
P_3 &= \{C' \to C, D' \to D\}, \\
P_4 &= \{C' \to S_R^{(3)}, D' \to S_R^{(2)}\}, \\
P_5 &= \{A^{(2)} \to B^{(2)} h''(u) \mid A, B \in N_R, u \in T, A \to Bu \in P_R\} \cup \\
&\quad \{A^{(2)} \to S_R^{(1)} h''(u) \mid A \in N_R, u \in T, A \to u \in P_R\} \cup \\
&\quad \{A^{(1)} \to B^{(1)} h'(u) \mid A, B \in N_R, u \in T, A \to Bu \in P_R\} \cup \\
&\quad \{A^{(1)} \to S_R h'(u) \mid A \in N_R, u \in T, A \to Bu \in P_R\}, \\
P_6 &= \{X' \to X, X^{(4)} \to X^{(3)} \mid X \in N_R\}, \\
Q_i &= \begin{cases}
\{A \to u^R B' u, A^{(3)} \to B^{(4)} u \mid A, B \in N_R, u \in T\} \cup \{S_R^{(2)} \to E\} \\
\quad \text{wenn } p_i = A \to Bu \in P_R, \\
\{A \to u^R u, A^{(3)} \to u \mid A \in N_R, u \in T\} \cup \{S_R^{(2)} \to E\} \\
\quad \text{wenn } p_i = A \to u \in P_R.
\end{cases}
\end{aligned}$$

für $1 \leq i \leq p$.

Das CD Grammatiksystem Γ_2 arbeitet im t-Modus. Zuerst wird mit P_1 die Satzform $C^{m-1}D$ eingeführt. Danach erzeugen P_2 und P_3 zu jedem Nichtterminal eine gleiche Anzahl von Symbolen a und b. Man erhält die Satzform

$(a^nC'b^n)^{m-1}a^nD'b^n$. Mit P_4 wird danach die Erzeugung von Symbolen a und b abgeschlossen. Außerdem werden die Nichtterminale für die Erzeugung von x, y und z eingeführt. Die aktuelle Satzform ist dann $(a^nS_R^{(3)}b^n)^{m-1}a^nS_R^{(2)}b^n$. Da in den Komponenten Q_i eine Produktion $S_R^{(2)} \to E$ enthalten ist, die jede Ableitung blockiert, die $S_R^{(2)}$ enthält, muß jetzt P_5 angewendet werden. P_5 arbeitet wie die Produktionen in (7.1), (7.2), (7.3) und (7.4) und man erhält $(a^nS_R^{(3)}b^n)^{m-1}a^nS_Ryzb^n$. Danach werden jeweils eine Komponente Q_i mit $1 \leq i \leq p$ und P_6 im Wechsel angewendet und dabei die Strings x und x^Rx erzeugt. Durch die unterschiedliche Markierung der Nichtterminale auf der rechten und linken Seite der Produktionen wird gewährleistet, daß immer genau eine Produktion an jedem Nichtterminal in der Satzform angewendet wird. Insgesamt arbeiten die Produktionen in Q_i aber wie die Produktionen in (7.5) und (7.6). Abschließend erhalten wir, daß $L(\Gamma_2, t) = L_2$. Außerdem sind in jeder Satzform nur höchstens m Nichtterminale enthalten. Der Index von Γ_2 ist also m. Nach Lemma 7.3 gibt es auch für die anderen Ableitungsmodi CD Grammatiksysteme vom Index m, die L_2 erzeugen.

Wenn $T(M)$ unendlich ist, dann ist nach Lemma 2.8 und Lemma 2.10 die Sprache *INVALC*$(M) \in \mathcal{L}(LIN) \setminus \mathcal{L}(REG)$, und $L' = \{yz \mid y \in h'(INVALC(M), z \in h''(INVALC(M))\}$ ist nach Lemma 7.9 eine 2-lineare Sprache. Wir nehmen jetzt an, daß es ein CD Grammatiksystem $\Gamma \in (CD\text{-}mFIN, =2)$ gibt, das L_2 erzeugt. Sei $\Gamma = (N, T, P_1, \ldots, P_n, S)$, r die maximale Anzahl von Terminalen in einer Produktion von Γ und l die Anzahl der Nichtterminale in N. Für jedes Wort $(a^ixb^i)^{m-1}a^ix^Rxyzb^i$ mit $i \geq 1$ in L_2 gibt es eine Ableitung D, deren Satzformen höchstens m Nichtterminale enthalten. Bei jeder Benutzung einer Komponente können höchstens $2r$ Terminale erzeugt werden. Wenn jetzt $i > 2r \cdot m \cdot (l+1)$ ist, wird in der Ableitung der Strings a^ib^i mehr als $m \cdot (l+1)$ mal die aktive Komponente gewechselt. Da es nur $m \cdot (l+1)$ verschiedene Abfolgen von m oder weniger Nichtterminalen gibt, kommt eine dieser Abfolgen doppelt in der Ableitung vor. Weiterhin müssen auch genau m Nichtterminale in diesem Teilstück der Ableitung vorhanden sein, da sonst durch mehrfache Anwendung dieses Teilstücks ein Wort erzeugt werden könnte, das nicht in L_2 ist. Es gibt also in der Ableitung D einen Abschnitt

$$\begin{aligned} S \quad &\overset{=2}{\Longrightarrow} \ldots \overset{=2}{\Longrightarrow} a^{i-q}A_1b^{i-q} \ldots a^{i-q}A_mb^{i-q} \overset{=2}{\Longrightarrow} \ldots \\ &\overset{=2}{\Longrightarrow} a^{i-q'}A_1b^{i-q'} \ldots a^{i-q'}A_mb^{i-q'} \ldots \overset{=2}{\Longrightarrow} (a^ixb^i)^{m-1}a^ix^Rxyzb^i \end{aligned}$$

mit $0 \leq q' < q \leq i$. Die Bezeichnung der Komponenten wurde hier bewußt weggelassen, da sie für die weitere Argumentation nicht wichtig sind.

Weiterhin muß es auf dem der Ableitung D entsprechenden Ableitungsbaum einen Pfad vom ersten A_j zum zweiten A_j für alle $1 \leq j \leq m$ geben, da man sonst durch mehrmaliges Anwenden des Abschnitts der oben angegebenen Ableitung, der jeweils das erste A_j in das zweite A_j überführt, ein Wort erzeugen könnte, das nicht in L_2 ist.

Mit den gleichen Argumenten wie oben kann man feststellen, daß es für ge-

nügend große x auf der Ableitung D einen Abschnitt

$$
\begin{aligned}
S \quad &\overset{=2}{\Longrightarrow} \ldots \overset{=2}{\Longrightarrow} a^i x_1 X_1 x_1' b^i \ldots \\
&\quad a^i x_1 X_{m-1} x_1' b^i a^i x_1'^R x_1^R X_m x_1 x_1' yz b^i \\
&\overset{=2}{\Longrightarrow} \ldots \overset{=2}{\Longrightarrow} a^i x_1 x_2 X_1 x_2' x_1' b^i \ldots \\
&\quad a^i x_1 x_2 X_{m-1} x_2' x_1' b^i a^i x_1'^R x_1^R x_2'^R x_2^R X_m x_2 x_2' x_1 x_1' yz b^i \\
&\overset{=2}{\Longrightarrow} \ldots \overset{=2}{\Longrightarrow} a^i x_1 x_2 x_3 X_1 x_3' x_2' x_1' b^i \ldots \\
&\quad a^i x_1 x_2 x_3 X_{m-1} x_3' x_2' x_1' b^i a^i x_1'^R x_1^R x_2'^R x_2^R x_3'^R x_3^R X_m x_3 x_3' x_2 x_2' x_1 x_1' yz b^i
\end{aligned}
$$

mit $x_1 x_2 x_3 x_3' x_2' x_1' = x$ und $|x_2 x_2'| > 0$ gibt.

Als nächstes können wir schließen, daß auf dem der Ableitung D entsprechenden Ableitungsbaum jeweils ein Pfad vom ersten X_j zum zweiten X_j mit $1 \leq j \leq m$ existiert, da sonst Wörter erzeugt werden könnten, die nicht in L_2 sind.

Weiterhin muß es einen Pfad vom zweiten A_j zum ersten X_j für $1 \leq j \leq m$ auf dem der Ableitung D entsprechendem Ableitungsbaum geben. Andernfalls wäre es nicht möglich, den jeweiligen String x zwischen a^i und b^i zu erzeugen.

Eine schematische Darstellung des angenommenen Ableitungsbaums, der der Ableitung D entspricht, ist in Abbildung 7.2 dargestellt.

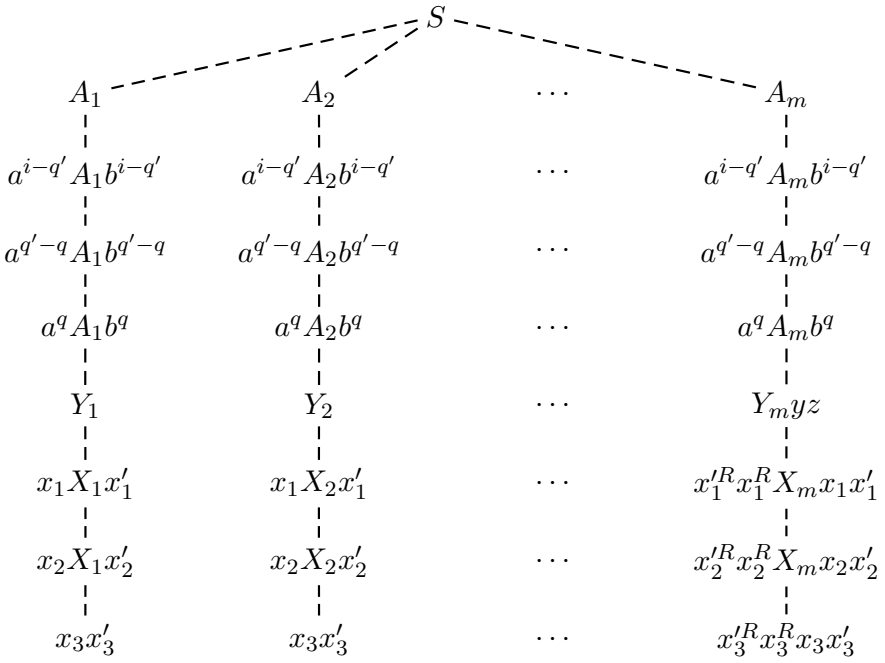

Abbildung 7.2: Schematische Darstellung einer angenommenen Ableitung mit nur m Nichtterminalen

Anhand der eben geführten Überlegungen kann man sehen, daß bei den Produktionen, die zu den betrachteten Ableitungen beitragen, die Terminale zur

Erzeugung von yz stets rechts vom letzten Nichtterminals auf der rechten Seite stehen müssen.

Wir können nun aus dem angenommenen CD Grammatiksystem Γ, das vom Index m ist, eine lineare Grammatik für $L' = \{yz \mid y \in h'(\mathit{INVALC}(M)), z \in h''(\mathit{INVALC}(M))\}$ konstruieren, was zu einem Widerspruch führt, da L'_2 2-linear ist.

Wir konstruieren zuerst $\Gamma' = (N, T, P'_1, \ldots, P'_n, S)$ mit $L(\Gamma', = 2) = L'$. Γ' erhalten wir auf einfache Weise aus Γ. Sei g der Homomorphismus mit $g(c) = c$ für alle $c \in N \cup h'(alph(\mathit{INVALC}(M))) \cup h''(alph(\mathit{INVALC}(M)))$ und $g(c) = \varepsilon$ für alle $c \in alph(\mathit{INVALC}(M)) \cup \{a, b\}$. $P'_i = \{A \to g(\alpha) \mid A \to \alpha \in P_i\}$ für $1 \leq i \leq n$.

Jetzt können wir aus dem CD Grammatiksystem Γ' das CD Grammatiksystem $\Gamma'' = (N'', T, P''_1, \ldots, P''_n, [S])$ konstruieren, das linear ist und für das auch gilt, daß $L(\Gamma'', = 2) = L'_2$. Γ'' arbeitet ähnlich wie Γ'. Der Unterschied ist, daß die m oder weniger Nichtterminale in einer Satzform zu einem Nichtterminal zusammengefaßt werden. Wir definieren für ein $E \notin N$ die Menge der Nichtterminale

$$N'' = \{[\beta] \mid \beta \in N^+, |\beta| \leq m\} \cup \{E\}$$

und für $1 \leq i \leq n$ sei

$$\begin{aligned}
P''_i \;=\; & \{[\gamma] \to [\delta]v \mid \gamma = \gamma' A, \delta = \gamma'\phi, A \to \phi v \in P'_i, \gamma, \delta \in N'', \\
& v \in (h'(alph(\mathit{INVALC}(M))) \cup h''(alph(\mathit{INVALC}(M))))^*\} \cup \\
& \{[\gamma] \to v \mid \gamma = \gamma' A, A \to v \in P'_i, \gamma \in N'', \\
& v \in (h'(alph(\mathit{INVALC}(M))) \cup h''(alph(\mathit{INVALC}(M))))^*\} \cup \\
& \{[\gamma] \to E \mid \gamma = \gamma' A\gamma'', \delta = \gamma'\phi\gamma'', A \to \phi v \in P'_i, \gamma \in N'', \delta \notin N'', \\
& v \in (h'(alph(\mathit{INVALC}(M))) \cup h''(alph(\mathit{INVALC}(M))))^*\} \cup \\
& \{[\gamma] \to E \mid \gamma = \gamma' A\gamma'', A \to v_0 A_1 v_1 A_2 v_2 \ldots v_{r-1} A_r v_r \in P'_i, \\
& \gamma \in N'', |v_0 v_1 \ldots v_{r-1}| \geq 1, \\
& v_0, \ldots, v_r \in (h'(alph(\mathit{INVALC}(M))) \cup h''(alph(\mathit{INVALC}(M))))^*\}.
\end{aligned}$$

In P''_i gibt es vier Arten von Produktionen. Die ersten beiden Arten von Produktionen arbeiten genauso wie die entsprechenden Produktionen aus Γ'. Der einzige Unterschied zwischen den erreichten Satzformen ist, daß die Nichtterminale, die in einer Satzform von Γ' unabhängig voneinander sind, in der korrespondierenden Satzform von Γ'' zu einem Nichtterminal zusammengefaßt sind. Die beiden anderen Arten von Produktionen, die in Γ'' durch das Nichtterminal E eine blockierte Satzform erreichen und in Γ' eventuell ein Wort erzeugen, sind in zwei Gruppen aufgeteilt. Die erste Sorte würde in Γ' zu einer Satzform mit Index größer m führen. Laut Definition des Index existiert aber für jedes Wort in $L(\Gamma', = 2)$ eine Ableitung, bei der alle Satzformen vom Index m oder kleiner sind.

Die zweite Sorte von Produktionen, die in Γ'' eine blockierte Satzform erreichen und in Γ' eventuell zur Erzeugung eines Wortes beitragen würden, ist

nicht linkslinear. Laut den vorherigen Überlegungen existieren aber für jedes Wort $yz \in L'$ auch Ableitungen in Γ'', bei denen alle Produktionen mit Terminalen linkslinear sind. Das ist der Fall, weil in der Grammatik Γ, aus der Γ' und dann Γ'' konstruiert wurden, für jedes $yz \in L'$ Wörter $(a^n x b^n)^{m-1} a^n x^R xyz b^n$ erzeugt werden können, bei denen die Teilwörter a^n, b^n, und x groß genug sind, um der vorher geführten Argumentation zu folgen.

Insgesamt ist $L(\Gamma', = 2) = L(\Gamma'', = 2) = L'$. Das führt zu einem Widerspruch, da Γ'' linkslinear, die Sprache L' aber nach Lemma 7.9 2-linear ist. Unsere Annahme, daß $\Gamma \in (CD\text{-}mFIN, = 2)\ L_2$ erzeugt, ist also falsch. Gäbe es jetzt ein CD Grammatiksystem in $(CD\text{-}mFIN, f)$, das L_2 erzeugt, dann könnte man nach Lemma 7.3 auch ein CD Grammatiksystem konstruieren, das im $(= 2)$-Modus arbeitet und L_2 erzeugt, was wiederum zum Widerspruch führt. Also ist $L_2 \notin \mathcal{L}(CD\text{-}mFIN, f)$. □

Lemma 7.11 *Sei M eine Turingmaschine und sei L_2 definiert wie in Lemma 7.10, dann ist für $f \in \{t, = k, \geq k \mid k \geq 2\}$ und $m \geq 2$*

$$L_2 \in \mathcal{L}(CD\text{-}(m+1)FIN, f).$$

Beweis: Für den Beweis werden wir ein $\Gamma \in (CD\text{-}(m+1)FIN, f)$ konstruieren, das L_2 erzeugt. Die Idee der Konstruktion kann wie folgt kurz zusammengefaßt werden. In der ersten Phase der Ableitung eines Wortes $w \in L_2$

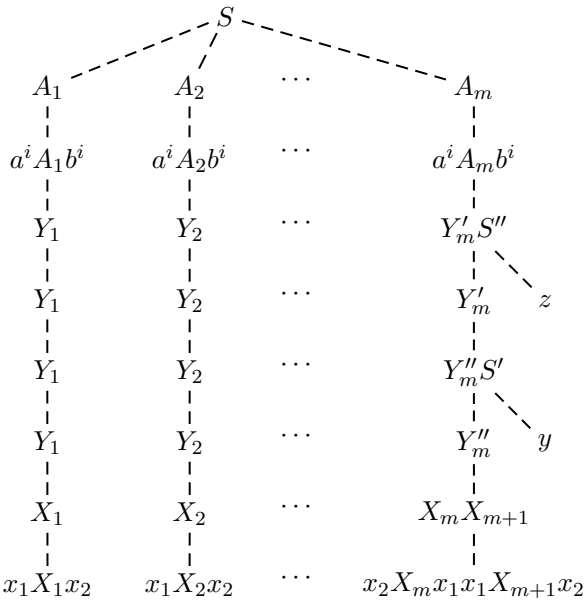

Abbildung 7.3: Schematisch dargestellter Ableitungsbaum von Γ

in Γ sind m Nichtterminale $A_1 \ldots A_m$ in der Satzform zu finden. Jedes dieser Nichtterminale generiert zwei Strings a^n und b^n, und wir erhalten die Satzform $a^n Y_1 b^n \ldots a^n Y_m b^m$. Danach wird das Nichtterminal Y_m durch $Y'_m S''$ ersetzt, und der String z wird erzeugt. Als nächstes ersetzt Γ das Nichtterminal Y'_m mit $Y''_m S'$ und erzeugt den String y. In der letzten Phase der Ableitung wird

Y_i mit X_i für $1 \leq i \leq m-1$ und Y''_m mit $X_m X_{m+1}$ überschrieben. Wir haben jetzt die Satzform $a^n X_1 b^n \ldots a^n X_m X_{m+1} yz b^m$. Mit den Nichtterminalen X_j, $1 \leq j \leq m-1$, und X_{m+1} generiert Γ jeweils einen String x, und X_m erzeugt x^R. Insgesamt erhalten wir $a^n x b^n \ldots a^n x^R xyz b^m$. Eine schematisch dargestellte Ableitung in Abbildung 7.3 verdeutlicht die Vorgehensweise des Grammatiksystems Γ noch einmal.

Da *INVALC*(M) eine lineare Sprache ist, können wir eine lineare Grammatik $G = (N, T, P, S)$ mit $T = alph(\mathit{INVALC}(M))$ und $L(G) = \mathit{INVALC}(M)$ finden. Dabei sei $P = \{p_1, \ldots, p_p\}$ die Menge der Produktionen. Weiterhin seien $G' = (N', T', P', S')$ und $G'' = (N'', T'', P'', S'')$ zwei disjunkte Kopien von G, die durch die Anwendung der Homomorphismen $h' : N \cup T \to N' \cup T'$, $h'(a) = a'$, $a \in N \cup T$ und $h'' : N \cup T \to N'' \cup T''$, $h''(a) = a''$, $a \in N \cup T$ entstehen. Im Gegensatz zu den in Kapitel 2 definierten Homomorphismen wirken sich h' und h'' auch auf die Terminale aus. Weiterhin seien $G^{(i)} = (N^{(i)}, T, P^{(i)}, S^{(i)})$, $1 \leq i \leq 4$ vier Kopien von G, bei denen nur die Nichtterminale markiert sind. Diese Kopien erhalten wir durch die Homomorphismen (i) für $1 \leq i \leq 4$, die wie im Kapitel 2 angegeben definiert sind.

Seien jetzt die Symbole $S_\Gamma, A_1, \ldots, A_m, Y_1, \ldots, Y_m, Y'_m, Y''_m, X_1, \ldots, X_{m+1}, F \notin N$ neue Nichtterminale. Das CD Grammatiksystem zur Erzeugung von L_2 ist wie folgt aufgebaut:

$$
\begin{aligned}
\Gamma &= (N_\Gamma, \{a,b\}, P_1, P_2, P_3, P_4, P_5, P_6, P_7, Q_1, \ldots, Q_{p+1}, S_\Gamma), \\
N_\Gamma &= \{S_\Gamma, A_1, \ldots, A_m, Y_1, \ldots, Y_m, Y'_m, Y''_m, X_1, \ldots, X_{m+1}, F\} \cup \\
&\quad N^{(1)} \cup \ldots \cup N^{(4)}, \\
P_1 &= \{S \to A_1 \ldots A_m\} \cup \{A'_i \to A_i \mid 1 \leq i \leq m\} \\
P_2 &= \{A_i \to a A'_i b \mid 1 \leq i \leq m\} \\
P_3 &= \{A_i \to Y_i \mid 1 \leq i \leq m\} \\
P_4 &= \{Y_m \to Y'_m S''\} \cup \{C \to \gamma \mid C \to \gamma \in P''\} \\
P_5 &= \{Y'_m \to Y''_m S'\} \cup \{C \to \gamma \mid C \to \gamma \in P'\} \\
P_6 &= \{Y_i \to X_i \mid 1 \leq i \leq m-1\} \cup \{Y''_m \to X_m X_{m+1}\} \cup \\
&\quad \{Y_m \to F, Y'_m \to F\} \\
P_7 &= \{X_i \to S^{(1)} \mid 1 \leq i \leq m-1\} \cup \{X_m \to S^{(3)}, X_{m+1} \to S^{(1)}\}, \\
Q_r &= \{D^{(1)} \to \alpha^{(2)}, D^{(3)} \to (\alpha^{(4)})^R \mid p_r = D \to \alpha\} \cup \\
Q_{p+1} &= \{E^{(2)} \to E^{(1)}, E^{(4)} \to E^{(2)} \mid E \in N\}.
\end{aligned}
$$

Das Grammatiksystem arbeitet im t-Modus und geht dabei vor, wie in der Beweisskizze erklärt. Eine Ableitung sieht wie folgt aus:

$$
\begin{aligned}
S_\Gamma \quad & \overset{t}{\Longrightarrow}_{P_1} A_1 \ldots A_m (\overset{t}{\Longrightarrow}_{P_2} \ldots \overset{t}{\Longrightarrow}_{P_1} \ldots)^i a^i A_1 b^i \ldots a^i A_m b^i \\
& \overset{t}{\Longrightarrow}_{P_3} a^i Y_1 b^i \ldots a^i Y_m b^i \\
& \overbrace{\Longrightarrow_{P_4} a^i Y_1 b^i \ldots a^i Y'_m S'' b^i \overset{*}{\Longrightarrow}_{P_4}}^{t} a^i Y_1 b^i \ldots a^i Y'_m z b^i
\end{aligned}
$$

$$
\begin{aligned}
&\overbrace{\Longrightarrow_{P_5} a^iY_1b^i \dots a^iY_m''S'zb^i \overset{*}{\Longrightarrow}_{P_5}}^{t} a^iY_1b^i \dots a^iY_m''yzb^i \\
&\overset{t}{\Longrightarrow}_{P_6} a^iX_1b^i \dots a^iX_mX_{m+1}yzb^i \\
&\overset{t}{\Longrightarrow}_{P_7} a^iS^{(1)}b^i \dots a^iS^{(3)}S^{(1)}yzb^i \\
&\overset{t}{\Longrightarrow}_{Q_{j_1}} \dots \overset{t}{\Longrightarrow}_{Q_{p+1}} \dots \overset{t}{\Longrightarrow}_{Q_{j_z}} \dots \overset{t}{\Longrightarrow}_{Q_{p+1}} a^ixb^i \dots a^ix^Rxyzb^i,
\end{aligned}
$$

dabei sind $z \geq 1$ und $1 \leq j_1, \dots, j_z \leq p$.

Zuerst muß Γ die Komponente P_1 benutzen, um $A_1 \dots A_m$ einzuführen. Danach können nur P_1 und P_2 im Wechsel angewendet werden, um vor bzw. hinter jedem Nichtterminal A_i, $1 \leq i \leq m$, eine gleiche Anzahl der Symbole a und b zu erzeugen. Nachdem beliebig viele Symbole a und b erzeugt wurden, wird die nächste Phase der Ableitung erreicht, indem P_3 benutzt wird. Wir erhalten $a^iY_1b^i \dots a^iY_mb^i$.

Danach erfolgt die Erzeugung von z. Dazu verwenden wir P_4. Mit der ersten Produktion wird das Startsymbol S'' eingeführt und mit den übrigen Produktionen ein String aus $h''(\mathit{INVALC}(M))$ erzeugt. Da die übrigen Produktionen Kopien der Produktionen von G sind, können sie nur Wörter aus $h''(\mathit{INVALC}(M))$ generieren. Außerdem ist durch den t-Modus gewährleistet, daß ein Terminalwort z erzeugt wird und kein Nichtterminal aus G'' in der Satzform verbleibt. Nach der Anwendung von P_4 ist die aktuelle Satzform $a^iY_1b^i \dots a^iY_m'zb^i$. Auf die gleiche Weise arbeitet danach P_5, und wir erhalten $Y_1b^i \dots a^iY_m''yzb^i$ mit einem Wort $y \in h'(\mathit{INVALC}(M))$.

Als nächstes kann nur P_6 angewendet werden. Mit den Produktionen $Y_m \to F$ und $Y_m' \to F$ wird erzwungen, daß vorher ein Wort yz erzeugt wurde. Insgesamt erhalten wir $a^iX_1b^i \dots a^iX_mX_{m+1}yzb^i$. Jetzt kann die Komponente P_7 aktiv werden, die jeweils X_i für $1 \leq i \leq m-1$ auf das Startsymbol $S^{(1)}$ der entsprechenden Kopie von G , X_m auf $S^{(3)}$ und X_{m+1} wiederum auf $S^{(1)}$ ableitet. Die aktuelle Satzform ist dann $a^iS^{(1)}b^i \dots a^iS^{(3)}S^{(1)}yzb^i$.

In der letzten Phase der Ableitung werden die Komponenten $Q_1, \dots, Q_p$ benutzt. Jede Komponente beinhaltet zwei Kopien einer Produktion aus G. So wird sichergestellt, daß von jedem Startsymbol $S^{(1)}$ der gleiche String bzw. von $S^{(3)}$ der entsprechende String gespiegelt erzeugt wird. Durch die zur linken Seite einer Produktion verschiedene Markierung der Nichtterminale auf der rechten Seite einer Produktion (2) und (4) wird sichergestellt, daß mit jeder Benutzung einer Komponente Q_r an jedem Nichtterminal genau eine Produktion angewendet wird. Q_{p+1} verändert diese Markierung dann wieder auf (1) beziehungsweise (3), und eine weitere Komponente Q_r kann angewendet werden. Am Ende der Ableitung erhalten wir ein Wort $a^ixb^i \dots a^ix^Rxyzb^i$. Dabei sind $i \geq 1$ und $x \in \mathit{INVALC}(M)$, $y \in h'(\mathit{INVALC}(M))$, $z \in h''(\mathit{INVALC}(M))$ beliebig, und die erzeugte Sprache ist L_2.

Da G und die verschiedenen Kopien dieser Grammatik linear sind, kann man sehen, daß die Anzahl der Nichtterminale in keiner Satzform die Zahl $m+1$ übersteigt. Es gilt also $L_2 \in \mathcal{L}(\mathit{CD}\text{-}(m+1)\mathit{FIN}, t)$. Nach Lemma 7.3 können

wir ein äquivalentes CD Grammatiksystem finden, das im Ableitungsmodus $f' \in \{= k, \geq k \mid k \geq 2\}$ arbeitet und auch vom Index $m+1$ ist. Es ist also $L_2 \in \mathcal{L}(\textit{CD-}(m+1)\textit{FIN}, f)$. □

Lemma 7.12 *Für jede Turingmaschine M und ein $c \notin alph(\textit{INVALC}(M))$ sei*

$$L_3 = (\textit{INVALC}(M)c)^*\textit{INVALC}(M).$$

Es gilt für jeden Ableitungsmodus $f \in \{t, = k, \geq k \mid k \geq 2\}$

$$L_3 \in \mathcal{L}(\textit{CD-METALIN}, f) \textit{ genau dann, wenn } T(M) \textit{ endlich ist.}$$

Beweis: Wenn $T(M)$ endlich ist, dann ist *VALC*(M) endlich und damit das Komplement *INVALC*$(M) \in \mathcal{L}(\textit{REG})$. Durch die Abschlußeigenschaften von $\mathcal{L}(\textit{REG})$ ist auch L_3 regulär und damit in $\mathcal{L}(\textit{CD-METALIN}, f)$.

Wenn $T(M)$ unendlich ist, dann ist nach Lemma 2.8 und Lemma 2.10 die Sprache *INVALC*$(M) \in \mathcal{L}(\textit{LIN}) \setminus \mathcal{L}(\textit{REG})$. Angenommen L_3 wäre in der Sprachklasse $\mathcal{L}(\textit{CD-METALIN}, f)$, dann wäre $L_3 \in \mathcal{L}(\textit{CD-}m\textit{LIN}, f)$ für ein $m \geq 1$. Sei jetzt

$$L_3' = L_3 \cap (alph(\textit{INVALC}(M))^*c)^{2m-1}alph(\textit{INVALC}(M))^*.$$

Da laut Satz 6.6 die Klasse $\mathcal{L}(\textit{CD-}m\textit{LIN}, f)$ unter Schnitt mit regulären Mengen abgeschlossen ist und $alph(\textit{INVALC}(M))^*c)^{2m-1}alph(\textit{INVALC}(M))^*$ regulär ist, wäre auch $L_3' \in \mathcal{L}(\textit{CD-}m\textit{LIN}, f)$. Das widerspricht aber Korollar 6.1. Also ist $L_3 \notin \mathcal{L}(\textit{CD-METALIN}, f)$. □

Lemma 7.13 *Für alle $f \in \{t, = k, \geq k \mid k \geq 2\}$ gilt*

$$L_3 \in \mathcal{L}(\textit{CD-FIN}, f).$$

Beweis: Wir konstruieren jetzt ein CD Grammatiksystem $\Gamma \in (\textit{CD-FIN}, t)$, das L_3 erzeugt. Da *INVALC*(M) eine lineare Sprache ist, gibt es eine lineare Grammatik $G = (N, alph(\textit{INVALC}(M)), P, S)$ mit $L(G) = \textit{INVALC}(M)$. Seien jetzt $S', B \notin N$. Das CD Grammatiksystem

$$\begin{aligned}
\Gamma &= (\{S', B\} \cup N, alph(\textit{INVALC}(M)) \cup \{c\}, P_1, P_2, S'), \\
P_1 &= \{S' \to ScB, S' \to S\}, \\
P_2 &= \{A \to \alpha \mid A \to \alpha \in P, \\
&\qquad \alpha \in alph(\textit{INVALC}(M))^*N^{\leq 1}alph(\textit{INVALC}(M))^*\} \cup \{B \to S'\}
\end{aligned}$$

erzeugt L_3 im t-Modus. Eine Ableitung in Γ benutzt zuerst die Komponente P_1, um S' durch ScB oder S zu ersetzen. Dann wird mit der Komponente P_2 und den Produktionen aus P von S aus ein Wort in *INVALC*(M) erzeugt. Falls B in der Satzform vorhanden ist, wird es mit S' ersetzt. Insgesamt sind immer nur zwei oder weniger Nichtterminale in einer Satzform vorhanden. Es werden Wörter aus *INVALC*(M) getrennt durch c erzeugt. Durch die Produktionen in P_1 und die Produktion $B \to S'$ in P_2 können beliebig viele dieser Wörter erzeugt werden. Insgesamt ist $L_3 = L(\Gamma, t)$. Nach Satz 6.3 können wir auch CD Grammatiksysteme mit gleichem Index konstruieren, die L_3 erzeugen und in einem Ableitungsmodus $f' \in \{= k, \geq k \mid k \geq 2\}$ arbeiten. □

Wir werden jetzt die Technik von Hartmanis aus [Har79] benutzen, die wir in Kapitel 2 eingeführt haben, um verschiedene nichtrekursive Tradeoffs zu beweisen. Dazu können wir Satz 2.7 auf die eben eingeführten Sprachen und deren Eigenschaften anwenden.

Satz 7.2 *Sei $f \in \{t, = k, \geq k \mid k \geq 2\}$, dann gilt*

$$(\textit{CD-mFIN}, f) \overset{nonrec}{\longleftarrow} (\textit{CD-}(m+1)\textit{FIN}, f)$$

für $m \geq 1$.

Beweis: Sei M eine Turingmaschine und sei L_2 definiert wie in Lemma 7.10, dann kann man nach Lemma 7.10 und Lemma 7.11 die Sprache L_2 aus M effektiv konstruieren, und es gilt $L_2 \in \mathcal{L}(\textit{CD-}(m+1)\textit{FIN}, f)$. Außerdem ist $L_2 \in \mathcal{L}(\textit{CD-mFIN}, f)$ genau dann, wenn $T(M)$ endlich ist. Mit Satz 2.7 erhalten wir das Ergebnis. □

Satz 7.3 *Sei $f \in \{t, = k, \geq k \mid k \geq 2\}$, dann gilt*

$$(\textit{CD-mLIN}, f) \overset{nonrec}{\longleftarrow} (\textit{CD-mFIN}, f)$$

für $m \geq 2$.

Beweis: Sei M eine Turingmaschine und L_1 definiert wie in Lemma 7.8, dann kann man nach Lemma 7.8 die Sprache L_1 aus M effektiv konstruieren, und es gilt $L_1 \in \mathcal{L}(\textit{CD-mFIN}, f)$. Nach Lemma 6.12 gilt auch $L_1 \in \mathcal{L}(\textit{CD-mLIN}, f)$ genau dann, wenn $T(M)$ endlich ist. Mit Satz 2.7 erhalten wir das obige Ergebnis. □

Satz 7.4 *Sei $f \in \{t, = k, \geq k \mid k \geq 2\}$, dann gilt*

$$(\textit{CD-METALIN}, f) \overset{nonrec}{\longleftarrow} (\textit{CD-FIN}, f).$$

Beweis: Sei M eine Turingmaschine und L_3 definiert wie in Lemma 7.12, dann ist nach Lemma 7.13 die Sprache L_3 aus M effektiv konstruierbar. Außerdem gilt $L_3 \in \mathcal{L}(\textit{CD-FIN}, f)$. Nach Lemma 7.12 ist darüber hinaus $L_3 \in \mathcal{L}(\textit{CD-METALIN}, f)$ genau dann, wenn $T(M)$ endlich ist. Mit Satz 2.7 erhalten wir das Ergebnis. □

Die Abbildung 7.4 zeigt die nichtrekursiven Tradeoffs, die wir in diesem Abschnitt erhalten haben. Kombiniert sind sie mit den nichtrekursiven Tradeoffs zwischen metalinearen CD Grammatiksystemen aus dem letzten Kapitel. Ein Pfeil repräsentiert dabei einen nichtrekursiven Tradeoff.

Korollar 7.1 *Sei $f \in \{t, = k, \geq k \mid k \geq 2\}$, dann ist es nicht entscheidbar, ob*

1. *eine Sprache aus $\mathcal{L}(\textit{CD-mFIN}, f)$ in $\mathcal{L}(\textit{CD-(m-1)FIN}, f)$ ist,*
2. *eine Sprache aus $\mathcal{L}(\textit{CD-mFIN}, f)$ in $\mathcal{L}(\textit{CD-mLIN}, f)$ ist,*
3. *eine Sprache aus $\mathcal{L}(\textit{CD-FIN}, f)$ in $\mathcal{L}(\textit{CD-METALIN}, f)$ ist.*

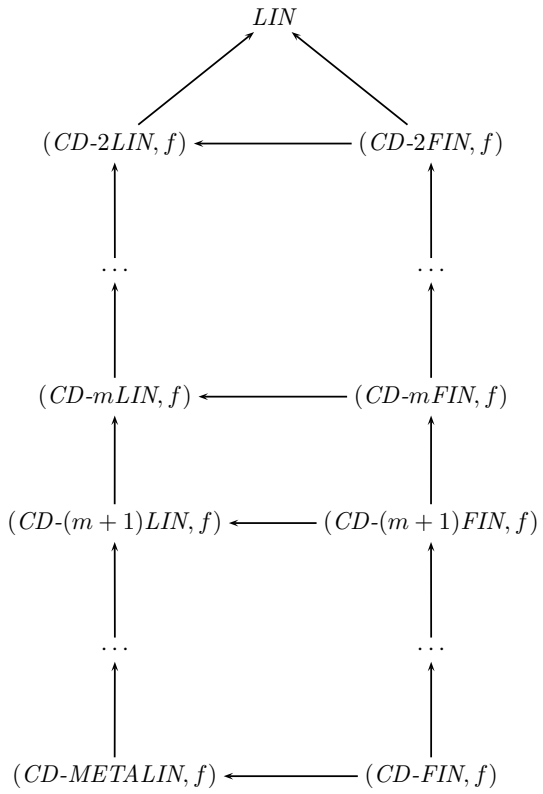

Abbildung 7.4: Nichtrekursive Tradeoffs bei metalinearen CD Grammatiksystemen und CD Grammatiksystemen von endlichem Index

Beweis: 1. Aus Lemma 7.10 und Lemma 7.11 kennen wir eine Sprache, die man für jede Turingmaschine M konstruieren kann. Diese Sprache ist in der Klasse $\mathcal{L}(CD\text{-}mFIN, f)$ und genau dann in der Klasse $\mathcal{L}(CD\text{-}(m-1)FIN, f)$, wenn $L(M)$ endlich ist. Könnte man also entscheiden, ob eine Sprache aus $\mathcal{L}(CD\text{-}mFIN, f)$ in $\mathcal{L}(CD\text{-}(m-1)FIN, f)$ ist, so könnte man entscheiden, ob $L(M)$ endlich ist. Das ist ein Widerspruch zu Satz 2.5.

2. Aus Lemma 7.8 und Lemma 6.12 kennen wir eine Sprache, die man für jede Turingmaschine M konstruieren kann. Diese Sprache ist in $\mathcal{L}(CD\text{-}mFIN, f)$ und genau dann in $\mathcal{L}(CD\text{-}mLIN, f)$, wenn $L(M)$ endlich ist. Das Ergebnis erhält man mit den gleichen Argumenten wie in 1.

3. Aus Lemma 7.12 und Lemma 7.13 kennen wir eine Sprache, die man für jede Turingmaschine M konstruieren kann. Diese Sprache ist in $\mathcal{L}(CD\text{-}FIN, f)$ und genau dann in $\mathcal{L}(CD\text{-}METALIN, f)$, wenn $L(M)$ endlich ist. Das Ergebnis erhält man mit den gleichen Argumenten wie in 1. □

7.4 Zusammenfassung

Nachdem wir im letzten Kapitel metalineare CD Grammatiksysteme untersucht haben, wurden in diesem Kapitel CD Grammatiksysteme von endlichem

Index betrachtet. Im Gegensatz zur sehr einfachen Struktur der metalinearen CD Grammatiksysteme, sind die CD Grammatiksysteme von endlichem Index nicht ganz so stark eingeschränkt.

CD Grammatiksysteme von endlichem Index haben eine größere generative Mächtigkeit als metalineare CD Grammatiksysteme. Auch bei ihnen besteht eine unendliche Hierarchie, allerdings nicht über der Breite, sondern über dem Index m. Außerdem existiert ein Pumpinglemma für die erhaltenen Sprachklassen, und die Abschlußeigenschaften entsprechen denen der metalinearen CD Grammatiksysteme, was auch $\mathcal{L}(\mathit{CD}\text{-}m\mathit{FIN}, f)$ zu einer vollen semi-AFL macht. Die generative Mächtigkeit im Vergleich zu metalinearen CD Grammatiksystemen wird in Abbildung 7.1 zusammengefaßt.

Im Bereich der Beschreibungskomplexität konnten wir die Fragen beantworten, inwieweit sich die Beschreibung vergrößern muß, wenn man eine Sprachklasse mit einem CD Grammatiksystem mit Index $m+1$ bzw. mit Index m beschreibt. Außerdem haben wir beantwortet, wie sich die Größe einer Beschreibung ändert, wenn man eine Sprachklasse mit einem CD Grammatiksystem mit Index m bzw. mit einem m-linearen CD Grammatiksystem beschreibt. In beiden Fällen erhalten wir einen nichtrekursiven Tradeoff. Genau dargestellt sind die Tradeoffs in Abbildung 7.4.

Weiterhin haben wir gesehen, daß für alle bewiesenen nichtrekursiven Tradeoffs die Frage, ob eine Sprache, die von einem CD Grammatiksystem aus der stärkeren Klasse erzeugt wird, auch von einem CD Grammatiksystem in der schwächeren Klasse erzeugt werden kann, nicht entscheidbar ist.

Kapitel 8

Diskussion und Ausblick

In dieser Arbeit haben wir die Beschreibungskomplexität von CD Grammatiksystemen untersucht. Dabei spielen verschiedene Maße für die statische Komplexität oder Größe dieser CD Grammatiksysteme eine Rolle.

Im Falle von kontextfreien CD Grammatiksystemen war bereits bekannt, daß man $mProd$, die maximale Anzahl von Produktionen in einer Komponente, beim t-Modus nach oben beschränken kann. Weiterhin sind drei Komponenten genug, um alle Sprachen zu erzeugen, die man mit dem t-Modus erhalten kann. Das entspricht einer Beschränkung des Maßes Deg, der Anzahl der Komponenten. Hält man jetzt im t-Modus und beim $*$-Modus einen der beiden Parameter $mProd$ oder Deg fest, dann induziert der andere eine unendliche Hierarchie von Sprachklassen.

Weiterhin wurde beim Vergleich zwischen kontextfreien CD Grammatiksystemen und kontextfreien Grammatiken festgestellt, daß es keine rekursive Funktion gibt, die den Tradeoff beschränkt, der zwischen kontextfreien Grammatiken und kontextfreien CD Grammatiksystemen besteht. Das gilt für die Maße Var, $Prod$ und $cProd$. Für $Symb$ läßt sich nur ein etwas schwächeres Ergebnis erzielen.

Auf der Basis der bereits bekannten Ergebnisse stellte sich zunächst die Frage, ob bei hybriden CD Grammatiksystemen, einer Generalisierung von kontextfreien CD Grammatiksystemen, ähnliche Ergebnisse gezeigt werden können. Es ist bereits bekannt, daß vier Komponenten im Falle der Modi $\{t, *, \leq k, = k, \geq k \mid k \geq 1\}$ genug sind, um alle Sprachen zu erzeugen, die mit hybriden CD Grammatiksystemen in diesen Modi erzeugt werden können. Wir haben gezeigt, daß man höchstens fünf Komponenten benötigt, wenn man den Ableitungsmodus $\overline{t}$ hinzunimmt. Das Maß Deg kann also nach oben begrenzt werden. Darüber hinaus konnten wir die maximale Anzahl von Produktionen in einer Komponente $mProd$ auf sechs beschränken, wenn der t-Modus beteiligt ist. Da bei diesem Ergebnis auch die Modi $\overline{t}$, $= k$, $\geq k$ und $\leq k$ mit einbezogen sind, geht es über das Ergebnis für kontextfreie CD Grammatiksysteme hinaus. Außerdem konnten wir bei hybriden CD Grammatiksystemen eine Hierarchie von Sprachklassen über dem Parameter Deg beziehungsweise $mProd$ zeigen, wenn der jeweils andere Parameter konstant ist. Auch hier reicht das Ergebnis etwas weiter als im Fall der kontextfreien CD Grammatiksysteme, weil es für alle Ableitungsmodi gezeigt wurde.

Da hybride CD Grammatiksysteme eine Generalisierung von kontextfreien

CD Grammatiksystemen sind, gelten die durch keine Funktion beschränkten Tradeoffs zu kontextfreien Grammatiken auch in ihrem Fall.

Die Frage, ob man eine Schranke für das Maß $mProd$ finden kann, wenn der t-Modus nicht beteiligt ist, stellt sich weiterführend. Es ist allerdings eher unwahrscheinlich, daß man auf ähnliche Weise wie in dieser Arbeit auch Ergebnisse für hybride CD Grammatiksysteme erreichen kann, die den t-Modus nicht benutzen. Das folgt aus den Konstruktionen für den entsprechenden Beweis, die zum größten Teil ohne Komponenten im t-Modus nicht zu funktionieren scheinen. Außerdem ist offen geblieben, ob die gezeigten oberen Schranken bei der maximalen Anzahl von Produktionen in einer Komponente optimal sind. Die konstruierten CD Grammatiksysteme sind von ihrer Struktur her so komplex, daß auch nicht zu erwarten ist, daß man direkt beweisen kann, ob sie optimal bezüglich $mProd$ sind. Außerdem fehlen weitgehend Hilfsmittel, mit denen man zeigen kann, ob eine bestimmte Sprache in einer der definierten Sprachklassen enthalten ist. Es wird also nicht einfach sein, eine gute untere Schranke zu zeigen.

Schon in der Einleitung haben wir die Eigenschaften von Komplexitätsmaßen diskutiert, die der intuitiven Vorstellung eines Maßes für die Größe einer Beschreibung entsprechen. Während die Maße Deg und $mProd$ aber auch $Prod$, $cProd$ und Var jeweils eher die Komplexität der Struktur eines Systems beleuchten, können Maße wie $Symb$ die Größe des Systems darstellen. Ein wichtiges Ziel dieser Arbeit war es, genaue Aussagen über Maße zu machen, die die Größe oder Länge einer Beschreibung darstellen und nicht nur einzelne Aspekte der Komplexität beleuchten, da über solche Maße bis jetzt nur Ergebnisse im Vergleich mit kontextfreien Grammatiken bekannt waren. Wir haben auch gesehen, daß sogar im Vergleich zu kontextfreien Grammatiken mit dem Maß $Symb$ nur etwas schwächere Ergebnisse erzielt werden konnten als mit den übrigen Maßen, die gerade nicht der intuitiven Vorstellung eines Größenmaßes entsprechen. Um genaue Resultate zu erhalten, wurden CD Grammatiksysteme daher zuerst stark eingeschränkt. Da kontextfreie CD Grammatiksysteme mit regulären oder linearen Produktionen nur die regulären beziehungsweise linearen Sprachen erzeugen, untersuchten wir zuerst die metalinearen CD Grammatiksysteme. Sie bestehen aus metalinearen Produktionen, die von der Struktur her nur etwas komplexer als lineare Produktionen sind. Zwar können metalineare CD Grammatiksysteme nicht alle kontextfreien Sprachen erzeugen, aber dennoch viele interessante kontextsensitive Sprachen generieren.

Als erstes wurde dann die generative Mächtigkeit der erhaltenen Sprachklassen bestimmt. Dabei haben wir gezeigt, daß für ein festes m die m-linearen CD Grammatiksysteme in allen starken Ableitungsmodi $f \in \{t, = k, \geq k \mid k \geq 2\}$ die gleiche Sprachklasse erzeugen. Diese Klasse fällt auch mit der Klasse der von m-linearen ET0L-Systemen und m-linearen Matrixgrammatiken generierten Sprachen zusammen. Weiterhin existiert eine unendliche Hierarchie von Sprachklassen über der Breite m von metalinearen CD Grammatiksystemen. Darüber hinaus konnten wir für die betrachteten Sprachklassen ein Pumpinglemma zeigen und durch die Abschlußeigenschaften sehen, daß

$\mathcal{L}(CD\text{-}mLIN, f)$ eine volle semi-AFL ist. Die generative Mächtigkeit von metalinearen CD Grammatiksystemen kann also sehr genau charakterisiert werden.

Weiterhin war es uns möglich, einen Hilfssatz darüber zu zeigen, inwieweit konkatenierte lineare Sprachen von metalinearen CD Grammatiksystemen erzeugt werden können. Dieses Lemma ähnelt dem in [Gre66] für kontextfreie metalineare Sprachen gezeigten Lemma. Durch die komplexe Struktur von CD Grammatiksystemen im Gegensatz zu den einfacheren kontextfreien Grammatiken mußten aber andere Methoden zum Beweis verwendet werden. Abgesehen von der Verwendung in der Beschreibungskomplexität zeigt das Lemma auch, welche erweiterten Möglichkeiten zur Erzeugung von metalinearen Sprachen CD Grammatiksysteme im Gegensatz zu kontextfreien Grammatiken bieten. Während eine m-lineare kontextfreie Grammatik höchstens m konkatenierte lineare Sprachen darstellen kann, sind es im Falle von m-linearen CD Grammatiksystemen $2m - 1$ konkatenierte lineare Sprachen.

Mit dem eben beschriebenen Hilfssatz konnten wir als nächstes mehrere Ergebnisse auf dem Gebiet der Beschreibungskomplexität zeigen. Wir benutzen dabei eine Methode aus [Har79], um nichtrekursive Tradeoffs zwischen m- und $(m + 1)$-linearen CD Grammatiksystemen zu beweisen. Weitere nichtrekursive Tradeoffs existieren zwischen m-linearen CD Grammatiksystemen und m-linearen kontextfreien Grammatiken. Außerdem gibt es einen nichtrekursiven Tradeoff zwischen metalinearen CD Grammatiksystemen und uneingeschränkten kontextfreien CD Grammatiksystemen. Diese Tradeoffs gelten für alle Maße $\mathcal{C}$, die der Voraussetzung entsprechen, daß es für jedes n nur endlich viele CD Grammatiksysteme Γ mit $\mathcal{C}(\Gamma) = n$ gibt. Wir haben damit eine Fülle von Ergebnissen für Maße erreicht, die man als vernünftige Größenmaße für CD Grammatiksysteme bezeichnen kann.

Es stellte sich dann die Frage, ob man ähnlich gute Ergebnisse auch für weniger stark eingeschränkte CD Grammatiksysteme erhalten kann. Für CD Grammatiksysteme von endlichem Index ist das der Fall. Hier war schon bekannt, daß CD Grammatiksysteme vom Index m für ein festes m in den starken Ableitungsmodi $f \in \{t, = k, \geq k \mid k \geq 2\}$ die gleiche Sprachklasse erzeugen. Diese Klasse fällt auch mit der Klasse der von ET0L-Systemen vom Index m und von Matrixgrammatiken vom Index m generierten Sprachen zusammen. Weiterhin existiert eine unendliche Hierarchie von Sprachklassen über dem Index m von kontextfreien CD Grammatiksystemen von endlichem Index. Darüber hinaus haben die betrachteten Sprachklassen ein Pumpinglemma und durch die Abschlußeigenschaften ist $\mathcal{L}(CD\text{-}mFIN, f)$ eine volle semi-AFL.

Ein ähnliches Lemma wie das über die formalen Eigenschaften von Sprachen, die von CD Grammatiksystemen der Breite m erzeugt werden können, konnte im Fall von CD Grammatiksystemen vom Index m nicht gezeigt werden. Für die Ergebnisse in der Beschreibungskomplexität war dann wieder die Methode aus [Har79] anwendbar. Allerdings mußten für jeden Tradeoff die geeigneten Sprachen gefunden werden. Der Nachweis der erforderlichen Eigenschaften dieser Sprachen mußte für jeden Fall gesondert geführt werden und war

teilweise sehr aufwendig. Trotzdem konnten ähnliche Ergebnisse wie im metalinearen Fall erreicht werden. Wir haben die Frage beantwortet, inwieweit sich die Beschreibung vergrößert, wenn man eine Sprachklasse mit einem CD Grammatiksystem mit Index $m + 1$ beziehungsweise mit Index m beschreibt. Außerdem haben wir beantwortet, wie sich die Größe einer Beschreibung ändert, wenn man eine Sprachklasse mit einem CD Grammatiksystem mit Index m beziehungsweise mit einem m-linearen CD Grammatiksystem beschreibt. In beiden Fällen erhalten wir einen nichtrekursiven Tradeoff. Diese Tradeoffs gelten für alle Maße $\mathcal{C}$, die der Voraussetzung entsprechen, daß es für jedes n nur endlich viele CD Grammatiksysteme Γ mit $\mathcal{C}(\Gamma) = n$ gibt. Wir haben damit auch im Falle des endlichen Index eine große Zahl an Ergebnissen für diese Art von Maßen erreicht.

Allerdings bleibt eine Frage in diesem Zusammenhang offen. Der Tradeoff zwischen CD Grammatiksystemen von endlichem Index und kontextfreien CD Grammatiksystemen konnte noch nicht bestimmt werden. Ein nichtrekursiver Tradeoff ist naheliegend, da es bestimmte kontextfreie Sprachen gibt, die nicht von CD Grammatiksystemen von endlichem Index erzeugt werden können. Mit Hilfe der Struktur dieser Sprache könnten sich weitere Sprachen konstruieren lassen, die man dann mit der Methode aus [Har79] zum Beweis eines nichtrekursiven Tradeoffs nutzen könnte.

Als nächster Schritt wäre es möglich, auf ähnliche Weise wie bei den bereits erhaltenen Ergebnissen, das Verhältnis zwischen kontextfreien Grammatiken von endlichem Index und CD Grammatiksystemen von endlichem Index zu untersuchen. Da keine allgemeingültigen Methoden dafür bekannt sind, müßten die dafür nötigen Sprachen jeweils gefunden und der Beweis für ihre Eigenschaften ad hoc geführt werden. Das Ergebnis ist vermutlich auch hier ein nichtrekursiver Tradeoff, da CD Grammatiksysteme im Gegensatz zu kontextfreien Grammatiken viele zusätzliche Funktionen zur Erzeugung von Sprachen haben, die man zum Beweis für diese nichtrekursiven Tradeoffs nutzen kann.

Eine Frage, die die Untersuchungen der Beschreibungskomplexität von CD Grammatiksystemen in eine andere Richtung lenken könnte, ist die nach der Existenz von rekursiven Tradeoffs. Wir haben zwar durch die Einschränkung von CD Grammatiksystemen auf Formen, die aus metalinearen Produktionen bestehen beziehungsweise von endlichem Index sind, viele Aussagen über Maße machen können, die den intuitiven Voraussetzungen an Größenmaße entsprechen. Allerdings haben wir auch gesehen, daß die erhaltenen nichtrekursiven Tradeoffs gewisse negative Eigenschaften mit sich bringen. Wenn wir zwei Sprachklassen betrachten, zwischen denen ein nichtrekursiver Tradeoff wie in [Har79] bewiesen werden kann, dann ist es nicht entscheidbar, ob eine Sprache, die in der stärkeren Klasse enthalten ist, auch in der schwächeren Klasse ist. Es stellt sich also unmittelbar die Frage, wie CD Grammatiksysteme beschaffen sein müssen, zwischen denen ein rekursiver Tradeoff besteht.

Ein Ansatzpunkt, um die eben gestellte Frage zu beantworten, wären deterministische CD Grammatiksysteme. Determinismus in einem Automatenmodell

bedeutet, daß es zu jedem Zeitpunkt nur eine Möglichkeiten geben kann, die Bearbeitung der Eingabe fortzusetzen. Im Gegensatz dazu gibt es keine allgemeingültige Definition für Determinismus in Grammatiken [Her99]. Ein erster Schritt wäre also, eine geeignete Definition für Determinismus bei CD Grammatiksystemen zu finden. Dabei ist zu beachten, daß CD Grammatiksysteme sowohl bei der Auswahl der nächsten aktiven Komponente als auch bei der nächsten verwendeten Produktion in einer Komponente nichtdeterministische Prozesse durchführen können. Zum ersten Mal werden deterministische CD Grammatiksysteme in [MM97] eingeführt, und es werden verschiedene Varianten diskutiert. Besonders sei aber auch auf eine Variante von CD Grammatiksystemen hingewiesen, die den kontextfreien $LL(k)$-Grammatiken ähnelt und in der Größenordnung $n \log^2 n$ analysiert werden kann. Der Begriff analysieren ist hier im Sinne des Parsing-Problems benutzt, welches darin besteht, festzustellen, ob ein gegebenes Wort von einer gegebenen Grammatik erzeugt wird. Wenn das so ist, soll eine Ableitung für dieses Wort ausgegeben werden. Bei einer kontextfreien $LL(k)$ Grammatik kann das Parsing-Problem in linearer Zeit gelöst werden, wenn man zu jedem Zeitpunkt des Parsing-Vorgangs die nächsten k Symbole des Eingabewortes kennt. Die Klasse der ähnlich definierten $LL(k)$ CD Grammatiksysteme wird in [BV04] ausführlich untersucht und würde sich für einen Einstieg in die Fragestellung eignen.

Auf diesem Weg sind nicht nur interessante Ergebnisse in der Beschreibungskomplexität zu erwarten, sondern man kann auch Klassen von CD Grammatiksystemen finden, durch deren gute Analyseeigenschaften auch eine Anwendung in der Praxis denkbar wäre.

Literaturverzeichnis

[BB93] Ludwig Balke und Karl Heinz Böhling. *Einführung in die Automatentheorie und Theorie formaler Sprachen*, Band 90 der *Reihe Informatik*. Bibliographisches Institut, Mannheim, 1993.

[BH00] Henning Bordihn und Markus Holzer. Grammar systems with negated conditions in their cooperation protocols. *J.UCS*, 6(12): 1165–1184, 2000.

[BR02] Henning Bordihn und Bernd Reichel. On descriptions of context-free languages by CD grammar systems. *J. Autom. Lang. Comb.*, 7 (4):447–454, 2002.

[BV04] Henning Bordihn und György Vaszil. CD grammar systems with LL(k) conditions. In Erzsébet Csuhaj-Varjú und György Vaszil, editors, *Proceedings of Grammar Systems Week 2004*, Seiten 95–112, Budapest, Hungary, July 5 - 9 2004. MTA SZTAKI.

[CS63] Noam Chomsky und Marcel P. Schützenberger. The algebraic theory of context-free languages. In *Computer programming and formal systems*, Seiten 118–161. North-Holland, Amsterdam, 1963.

[CVD90] Erzsébet Csuhaj-Varjú und Jürgen Dassow. On cooperating/distributed grammar systems. *J. Inform. Process. Cybernet.*, 26(1-2): 49–63, 1990.

[CVDKP94] Erzsébet Csuhaj-Varjú, Jürgen Dassow, Jozef Kelemen und Gheorghe Păun. *Grammar Systems - a grammatical approach to distribution and cooperation*, Band 5 der Reihe *Topics in Computer Mathematics*. Gordon and Breach Science Publishers, Yverdon, 1994.

[DM96] Jürgen Dassow und Victor Mitrana. Fairness in grammar systems. *Acta Cybernet.*, 12(4):331–345, 1996.

[DP89] Jürgen Dassow und Gheorghe Păun. *Regulated rewriting in formal language theory*, Band 18 der Reihe *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1989.

[DP90] Jürgen Dassow und Gheorghe Păun. On some variants of cooperating/distributed grammar systems. *Stud. Cerc. Mat.*, 42(2): 153–165, 1990.

[DP91] Jürgen Dassow und Gheorghe Păun. On the succinctness of descriptions of context-free languages by cooperating/distributed grammar systems. *Comput. Artificial Intelligence*, 10(6):513–527, 1991.

[DPR97] Jürgen Dassow, Gheorghe Păun und Grzegorz Rozenberg. Grammar systems. In *Handbook of formal languages, Bd. 2*, Seiten 155–213. Springer, Berlin, 1997.

[DPV95] Jürgen Dassow, Gheorghe Păun und Sorina Vicolov. On the generative capacity of certain classes of cooperating grammar systems. *Fund. Inform.*, 22(3):217–226, 1995.

[ER77] Andrzej Ehrenfeucht und Grzegorz Rozenberg. On some context free languages that are not deterministic ET0L languages. *RAIRO Informat. Théor.*, 11(4):273–291, 1977.

[FHF01] Henning Fernau, Markus Holzer und Rudolf Freund. Hybrid modes in cooperating distributed grammar systems: internal versus external hybridization. *Theoret. Comput. Sci.*, 259(1-2):405–426, 2001.

[GKK+02] Jonathan Goldstine, Martin Kappes, Chandra M. R. Kintala, Hing Leung, Andreas Malcher und Detlef Wotschke. Descriptional complexity of machines with limited resources. *J.UCS*, 8(2):193–234, 2002.

[GP90] Marian Gheorghe und Gheorghe Păun. Further remarks on cooperating/distributed grammar systems. *Bull. Math. Soc. Sci. Math. R. S. Roumanie (N.S.)*, 34(3):231–245, 1990.

[GPW82] Jonathan Goldstine, John K. Price und Detlef Wotschke. A pushdown automaton or a context-free grammar - which is more economical? *Theoret. Comput. Sci.*, 18(1):33–40, 1982.

[GPW93] Jonathan Goldstine, John K. Price und Detlef Wotschke. On reducing the number of stack symbols in a PDA. *Math. Systems Theory*, 26(4):313–326, 1993.

[GPW82] Jonathan Goldstine, John K. Price und Detlef Wotschke. On reducing the number of states in a PDA. *Math. Systems Theory*, 15(4): 315–321, 1981/82.

[Gre66] Sheila A. Greibach. The unsolvability of the recognition of linear context-free languages. *J. Assoc. Comput. Mach.*, 13:582–587, 1966.

[Gru67] Josef Gruska. On classification of context-free languages. *Kybernetika (Prague)*, 3:22–29, 1967.

[Gru69] Josef Gruska. Some classifications of context-free languages. *Information and Control*, 14:152–179, 1969.

[Gru73] Josef Gruska. Descriptional complexity of context-free languages. In *Mathematical foundations of computer science (Proc. Sympos. and Summer School, Štrbské Pleso, 1973)*, Seiten 71–83. Math. Inst., Slovak Acad. Sci., Bratislava, 1973.

[Har67] Juris Hartmanis. Context-free languages and Turing machine computations. In *Proc. Sympos. Appl. Math., Bd. XIX*, Seiten 42–51. Amer. Math. Soc., Providence, R.I., 1967.

[Har79] Juris Hartmanis. On the succinctness of different representations of languages. In *Automata, languages and programming (Sixth Colloq., Graz, 1979)*, Band 71 der Reihe *LNCS*, Seiten 282–288. Springer, Berlin, 1979.

[Har80] Juris Hartmanis. On the succinctness of different representations of languages. *SIAM J. Comput.*, 9(1):114–120, 1980.

[Her99] Christian Herzog. *Die Rolle des Nichtdeterminismus in kontextfreien Sprachen*. Dissertation, Johann Wolfgang Goethe-Universität, Frankfurt am Main, 1999.

[HU79] John E. Hopcroft und Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Co., Reading, Mass., 1979.

[Mal] Andreas Malcher. On non-recursive trade-offs between finite-turn pushdown automata. *J. Autom. Lang. Comb.* zur Veröffentlichung akzeptiert.

[Mal02] Andreas Malcher. Descriptional complexity of cellular automata and decidability questions. *J. Autom. Lang. Comb.*, 7(4):549–560, 2002.

[Mal04] Andreas Malcher. *Beschreibungskomplexität von Zellularautomaten*. Dissertation, Johann Wolfgang Goethe-Universität, Frankfurt am Main, 2004.

[MF71] Albert R. Meyer und Michael J. Fischer. Economy of description by automata, grammars, and formal systems. In *IEEE Twelfth Annual Symposium on Switching and Automata Theory*, Seiten 188–191. IEEE, East Lansing, Michigan, October 13 - 15 1971.

[Mit93] Victor Mitrana. Hybrid cooperating/distributed grammar systems. *Comput. Artificial Intelligence*, 12(1):83–88, 1993.

[MM97] Valeria Mihalache und Victor Mitrana. Deterministic cooperating distributed grammar systems. In *New trends in formal languages*, Band 1218 der Reihe *LNCS*, Seiten 137–149. Springer, Berlin, 1997.

[MR78] Robert Meersman und Grzegorz Rozenberg. Cooperating grammar systems. In *Mathematical foundations of computer science, 1978 (Proc. Seventh Sympos., Zakopane, 1978)*, Band 64 der Reihe *LNCS*, Seiten 364–373. Springer, Berlin, 1978.

[Pău92] Gheorghe Păun. On the syntactic complexity of parallel communicating grammar systems. *Kybernetika (Prague)*, 28(2):155–166, 1992.

[Pău94] Gheorghe Păun. On the generative capacity of hybrid CD grammar systems. *J.Inform.Process.Cybernet.*, 30(4):231–244, 1994.

[PDS94] Gheorghe Păun, Jürgen Dassow und Stefan Skalla. On the size of components of cooperating grammar systems. In *Results and trends in theoretical computer science (Graz, 1994)*, Band 812 der Reihe *LNCS*, Seiten 325–343. Springer, Berlin, 1994.

[Pir75] Alica Pirická. Complexity and normal forms of context-free languages. In *Mathematical foundations of computer science, 1974 (Jadwisin near Warsaw, 1974)*, Band 28 der Reihe *LNCS*, Seiten 292–297. Springer-Verlag, Berlin, 1975.

[PK75] Alica Pirická-Kelemenová. Greibach normal form complexity. In *Mathematical foundations of computer science (Fourth Sympos., Marlánské Lázně, 1975)*, Band 32 der Reihe *LNCS*, Seiten 344–350. Springer-Verlag, Berlin, 1975.

[Ros69] Daniel J. Rosenkrantz. Programmed grammars and classes of formal languages. *J. Assoc. Comput. Mach.*, 16:107–131, 1969.

[RS80] Grzegorz Rozenberg und Arto Salomaa. *The mathematical theory of L systems*, Band 90 der Reihe *Pure and Applied Mathematics*. Academic Press Inc., New York, 1980.

[RV78a] Grzegorz Rozenberg und Dirk Vermeir. On ET0L systems of finite index. *Inform. and Control*, 38(1):103–133, 1978.

[RV78b] Grzegorz Rozenberg und Dirk Vermeir. On the effect of the finite index restriction on several families of grammars. *Inform. and Control*, 39(3):284–302, 1978.

[RV80] Grzegorz Rozenberg und Dirk Vermeir. On metalinear ET0L systems. *Fund. Inform. (4)*, 3(1):15–36, 1980.

[Sal78] Arto K. Salomaa. *Formale Sprachen*. Springer-Verlag, Berlin, 1978.

[Suna] Bettina Sunckel. On the descriptional complexity of CD grammar systems of finite index. *J. Autom. Lang. Comb.* zur Veröffentlichung akzeptiert.

[Sunb] Bettina Sunckel. On the descriptional complexity of external hybrid cooperating distributed grammar systems. *J. Autom. Lang. Comb.* zur Veröffentlichung akzeptiert.

[Sun03] Bettina Sunckel. On the descriptional complexity of external hybrid cooperating distributed grammar systems. In Erzsébet Csuhaj-Varjú, Chandra Kintala, Detlef Wotschke, and György Vaszil, editors, *Proceedings of the Fifth International Workshop on Descriptional Complexity of Formal Systems 2003*, Seiten 198–209, Budapest, Hungary, July 12 - July 14 2003. MTA SZTAKI.

[Sun05a] Bettina Sunckel. A note on the descriptional complexity of CD grammar systems of finite index. In Carlo Mereghetti, Giovanni Pighizzini und Detlef Wotschke, editors, *Proceedings of the Seventh International Workshop on Descriptional complexity of formal systems*

2005, Seiten 215–226, Como, Italy, June 30 - July 2 2005. Università degli Studi di Milano.

[Sun05b] Bettina Sunckel. On the descriptional complexity of metalinear CD grammar systems. *Internat. J. Found. Comput. Sci.*, 16(5):1011–1025, 2005.

[Sun07] Bettina Sunckel. On metalinear CD grammar systems. *Fund. Inform.*, 76(3):383–397, 2007.

[vS76] Sebastiaan H. von Solms. Some notes on ET0L-languages. *Internat. J. Comput. Math.*, 5(4):285–296, 1975/76.

Lebenslauf

Name:	Bettina Sunckel
Geburtstag:	22. März 1971
Geburtsort:	Frankfurt am Main
Familienstand:	ledig
Wohnort:	Sophienstraße 51, 60487 Frankfurt am Main
Schulbildung:	
1977 – 1981	Grundschule, Westerbachschule in Eschborn
1981 – 1990	Gymnasium, St. Angela Schule in Königstein
Hochschulstudium:	
1990 – 2001	Studium der Informatik mit Nebenfach Mathematik an der J. W. Goethe–Universität Frankfurt am Main
Juli 2001	Abschluß als Diplom-Informatikerin an der J. W. Goethe–Universität Frankfurt am Main Diplomarbeit bei Prof. Dr. Detlef Wotschke
Berufstätigkeit:	
1997 – 2001	Freiberufliche Tätigkeit als Softwareentwicklerin und IT-Beraterin in Teilzeit
2001 – 2006	Wissenschaftliche Mitarbeiterin an der Professur Programmiersprachen und Compiler, Institut für Informatik, J. W. Goethe–Universität Frankfurt am Main, bei Prof. Dr. Detlef Wotschke
Seit 2007	Softwareentwicklerin im Hochschulrechenzentrum der J. W. Goethe–Universität Frankfurt am Main

Zusammenfassung

In der klassischen Theorie der formalen Sprachen gehört die Beschreibung von Sprachen durch Grammatiken oder Automaten zu den wichtigen Themen. Im Gegensatz zu diesen Modellen, die aus einer einzelnen Komponente bestehen, beschäftigt sich die Informatik heute aber immer häufiger mit verteilten Systemen, deren Komponenten auf verschiedene Art und Weise zusammenarbeiten. Eine Möglichkeit, dieses Konzept auf die Theorie der formalen Sprachen zu übertragen, ist die Definition von Grammatiksystemen. Ein Grammatiksystem besteht aus mehreren Grammatiken, die nach bestimmten Regeln zusammenarbeiten. Hauptsächlich unterscheidet man dabei zwischen sequentieller und paralleler Kooperation. In dieser Arbeit werden kontextfreie „cooperating distributed“ (CD) Grammatiksysteme, ein Modell mit sequentieller Kooperation, betrachtet. Zur Erzeugung eines Wortes arbeiten dabei mehrere kontextfreie Grammatiken, die Komponenten, an einer gemeinsamen Satzform. Zu jedem Zeitpunkt ist immer nur eine einzige Komponente aktiv.

Der Schwerpunkt der Arbeit liegt auf der Beschreibungskomplexität von CD Grammatiksystemen. Dabei wird zuerst auf die verschiedenen Maße für die Größe oder statische Komplexität eines CD Grammatiksystems eingegangen. Ein wichtiges Ergebnis im ersten Teil der Arbeit ist, daß man für CD Grammatiksysteme und insbesondere hybride CD Grammatiksysteme, eine Verallgemeinerung von kontextfreien CD Grammatiksystemen, einige dieser Maße nach oben beschränken kann. Darunter fallen die Anzahl der Komponenten und die maximale Anzahl von Produktionen in einer Komponente. Hält man einen der beiden Parameter fest, so entsteht eine unendliche Hierarchie über dem anderen Parameter.

Der zweite Teil der Arbeit konzentriert sich darauf, Ergebnisse für Größenmaße zu erzielen, die nicht nur einzelne Aspekte der Komplexität, sondern die gesamte Größe oder Länge eines CD Grammatiksystems darstellen. Dafür werden CD Grammatiksysteme geeignet eingeschränkt. Man erhält metalineare Systeme und Systeme von endlichem Index. Im Gegensatz zum unbeschränkten Modell kann hier die generative Mächtigkeit sehr genau charakterisiert werden und es können Hilfsmittel wie Pumpinglemmata gezeigt werden. Weitere Resultate sind eine unendliche Hierarchie über der Breite beziehungsweise dem Index solcher Grammatiksysteme.

Das wesentliches Resultat im zweiten Teil dieser Arbeit besteht daraus, daß zwischen zwei Klassen von diesen eingeschränkten CD Grammatiksystemen, deren entsprechende Sprachklassen echt ineinander enthalten sind, nichtrekursive Tradeoffs existieren. Das heißt, daß sich der Größenzuwachs beim Wechsel von der stärkeren Klasse von CD Grammatiksystemen in die schwächere durch keine rekursive Funktion beschränken läßt.