

# Automatendefinierbare Baumrelationen mit Anzahlbedingungen

**Nils Jansen**  
Matrikelnummer 243216

**Diplomarbeit**  
im Studiengang Informatik

vorgelegt der Fakultät  
für Mathematik, Informatik und Naturwissenschaften

im Dezember 2008

angefertigt am  
Lehrstuhl für Informatik 7  
Prof. Dr. Dr.h.c. Wolfgang Thomas



## **Erklärung**

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, den 23. Dezember 2008

---

Nils Jansen



**Abstract**

The definition of *rational relations* on words and their properties are well-known in theoretical computer science. There are several extensions of these relations to ranked trees whereas the rational relations over unranked trees are an unregarded topic up to now. In this thesis we start with the introduction of an existing automata theoretic approach by which the resulting class of relations turns out to be a natural extension of the rational word relations. Furthermore, we introduce new models of automata recognizing exactly this class of relations and some restrictions, respectively. All important decision problems are considered.

In the field of XML research the document structure is determined by its DTD. In addition to this, there may be cases where a constraint on the cardinalities of certain elements of one document is needed. As XML documents form unranked trees, there are several approaches involving Presburger arithmetic or the Parikh map to constrain trees and therefore XML-documents. We use these approaches to define some classes of “counting” tree relations by combining them with our automata for rational tree relations. It turns out that all decidable problems of the rational word relations remain decidable for tree relations.

**Zusammenfassung**

*Rationale Relationen* über Wörtern sind in der theoretischen Informatik weitestgehend erforscht. Es existieren verschiedene Erweiterungen dieser Relationen auf beschränkt verzweigte Bäume, während rationale Relationen über unbeschränkt verzweigten Bäumen bislang nicht definiert sind. In dieser Arbeit beginnen wir mit der Einführung eines existierenden Automatenmodells, durch das sich die resultierende Klasse von Baumrelationen als eine natürliche Erweiterung der rationalen Wortrelationen herausstellt. Es werden neue Automaten vorgestellt, die genau diese neue Klasse von Relationen bzw. eine Einschränkung mit nützlichen Eigenschaften definieren. Alle wichtigen Entscheidungsprobleme werden betrachtet.

Gerade im Bereich der XML-Forschung erscheint es sinnvoll, zusätzlich zu den durch eine DTD gegebenen Strukturvorgaben für ein XML-Dokument Anzahlbedingungen an bestimmte Elemente des Dokuments stellen zu können, das formal betrachtet ein unbeschränkt verzweigter Baum ist. Es gibt verschiedene Ansätze, die mit Hilfe der Presburger Arithmetik oder der Parikh-Abbildung Anzahlen in Bäumen und damit auch in XML-Dokumenten beschränken können. Wir nutzen dies, um Anzahlbedingungen für Relationen von unbeschränkt verzweigten Bäumen zu formulieren. Dazu wird das hier vorgestellte Automatenmodell mit allen Ansätzen kombiniert und wir erhalten mehrere Klassen von „zählenden“ Baumrelationen. Es stellt sich heraus, dass alle entscheidbaren Probleme der rationalen Wortrelationen auch für die Baumrelationen entscheidbar sind.



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Grundlagen</b>	<b>9</b>
2.1	Automaten und formale Sprachen . . . . .	9
2.2	Wortrelationen . . . . .	10
2.3	Bäume und Baumautomaten . . . . .	12
<b>3</b>	<b>Anzahlbedingungen</b>	<b>17</b>
3.1	Semi-lineare Mengen und Presburger-Arithmetik . . . . .	17
3.2	Parikh-Zählen . . . . .	21
3.2.1	Parikh-Abbildung . . . . .	21
3.2.2	Parikh-Automat . . . . .	22
3.3	Baumautomaten mit lokalen Anzahlbedingungen . . . . .	25
3.3.1	Presburger-Baumautomaten . . . . .	26
3.3.2	Parikh-Baumautomaten . . . . .	29
3.4	Baumautomaten mit globalen Anzahlbedingungen . . . . .	31
3.4.1	Presburger-Baumautomaten mit globalen Anzahlbedingungen . . .	32
3.4.2	Parikh-Baumautomaten mit globalen Anzahlbedingungen . . . . .	33
3.5	Vergleich der Modelle . . . . .	34
<b>4</b>	<b>Rationale Baumrelationen</b>	<b>39</b>
4.1	Multivariablen . . . . .	40
4.2	Asynchrone Baumautomaten . . . . .	43
<b>5</b>	<b>Rationale Relationen über unbeschränkt verzweigten Bäumen</b>	<b>49</b>
5.1	Asynchrone unbeschränkt verzweigte Baumautomaten . . . . .	50
5.1.1	Beziehung zu rationalen Wortrelationen . . . . .	58
5.1.2	Entscheidungsprobleme . . . . .	60
5.2	Synchronisierte Transitionen . . . . .	64
5.2.1	Äquivalenz zu asynchronen Baumautomaten . . . . .	70
5.2.2	Entscheidungsprobleme . . . . .	75

5.3	Transitions-Separierte Baumrelationen . . . . .	77
<b>6</b>	<b>Baumrelationen mit Anzahlbedingungen</b>	<b>81</b>
6.1	Lokale Anzahlbedingungen für Baumrelationen . . . . .	81
6.1.1	Presburger-Baumautomaten für Relationen . . . . .	81
6.1.2	Parikh-Baumautomaten für Relationen . . . . .	84
6.1.3	Vergleich lokal zählender Baumrelationen . . . . .	85
6.1.4	Entscheidungsprobleme . . . . .	87
6.2	Globale Anzahl-Bedingungen für Baumrelationen . . . . .	89
6.2.1	Global zählende Presburger-Baumautomaten für Relationen . . . . .	89
6.2.2	Global zählende Parikh-Baumautomaten für Relationen . . . . .	90
6.2.3	Vergleich global zählender Baumrelationen . . . . .	93
6.2.4	Entscheidungsprobleme . . . . .	94
6.3	Klassen von zählenden Baumrelationen . . . . .	96
<b>7</b>	<b>Zusammenfassung</b>	<b>99</b>

# Kapitel 1

## Einleitung

Das zentrale Thema dieser Arbeit sind Relationen und deren Eigenschaften unter dem Gesichtspunkt der Theorie der formalen Sprachen. Sprachen von endlichen Wörtern werden unter Berücksichtigung verschiedener allgemeiner Vorgaben in Verbindung zueinander gesetzt; diese Vorgaben führen zu verschiedenen Klassen von Relationen.

Wir geben zunächst ein kurzes Beispiel an, dabei ist die Relation  $R$  eine Teilmenge des kartesischen Produktes zweier regulärer Sprachen:

$$R = \{(u, vu) \mid u, v \in \Sigma^*\} \subseteq \Sigma^* \times \Sigma^* .$$

Diese Menge  $R$  enthält alle Wortpaare, für die das erste Wort ein Suffix des zweiten Wortes ist. Eine solche Relation gehört zu der Klasse der *rationalen Relationen*, mit denen wir uns ausschließlich beschäftigen werden. Die rationalen Relationen über endlichen Wörtern bilden ein intensiv erforschtes Themengebiet der theoretischen Informatik, und es herrscht ein allgemeiner Konsens über die Charakterisierung und Eigenschaften dieser Relationen. Es gibt verschiedene Formalismen, die rationale Relationen beschreiben. Sie können beispielsweise durch *rationale Ausdrücke* definiert werden, die eine Erweiterung der *regulären Ausdrücke* sind. Eine Relation wird ausgehend von einer endlichen Relation durch Abschluss unter Vereinigung, komponentenweiser Konkatenation und Kleene-Stern erzeugt. Ein anderer Ansatz geht von Automatenmodellen aus, sodass Automaten eines gewissen Typs genau die fraglichen Relationen charakterisieren. Solche Relationen werden *automatendefinierbar* genannt.

Rabin und Scott entwickelten 1959 ein deterministisches Automatenmodell, das eine Teilklasse der rationalen Relationen definiert [27]. Elgot und Mezei entwickelten 1965 ein mächtigeres, nichtdeterministisches Modell, das genau die Klasse der rationalen Relationen erkennt [11]. Sie zeigten wichtige Eigenschaften dieser Relationen auf, unter anderem die Abschlusseigenschaften. Die wichtigsten Entscheidungsprobleme wurden von Fischer und Rosenberg [13] betrachtet, sie lieferten unter anderem wichtige Unentscheidbarkeitsresultate. Ausführliche Einführungen zu diesen Relationen sind in den Büchern von

Berstel [4], Sakarovitch [32] und Eilenberg [10] nachzulesen. Um einen Überblick über die Historie und andere Klassen von Relationen über Wörtern zu erhalten, sei auf [6] verwiesen.

*Bäume* sind in der Informatik ein verbreiteter Formalismus unter anderem zur Darstellung und Beschreibung hierarchischer Strukturen. Im Gebiet der klassischen Sprachtheorie werden Baumautomaten zur Definition und Verifizierung von Baumsprachen verwendet. Ihre Eigenschaften sind weitgehend erforscht [17, 14].

Wir unterscheiden zwischen *beschränkt verzweigten* und *unbeschränkt verzweigten Bäumen*. Die beschränkt verzweigten Bäume sind über einem *Rangalphabet* definiert; für jeden Knoten ist durch den Rang seines Beschriftungssymbols festgelegt, wie viele Nachfolger er hat. Unbeschränkt verzweigte Bäume sind über einem reinen *Beschriftungsalphabet* definiert; die Anzahl der Nachfolger eines Knotens ist unbeschränkt, aber endlich. Diese Baumsprachen und *unbeschränkt verzweigte Baumautomaten* wurden vermutlich von Thatcher 1967 eingeführt [37, 38]. Da ein XML-Dokument einen unbeschränkt verzweigten Baum kodiert, wurden im Rahmen der XML-Forschung viele neue Erkenntnisse über *unbeschränkt verzweigte Baumautomaten* gewonnen [5, 21, 22].

Für einen ausführlichen Überblick über Historie, verschiedene Ansätze und Resultate sei auf das Buch *Tree Automata Techniques and Applications* verwiesen [7].

Wenn man sich praktische Anwendungen von XML, zum Beispiel Datenbanken oder einheitlichen Dokumentenaustausch, vor Augen führt, ist es denkbar, dass die Anzahl gewisser Knoten eingeschränkt werden soll. Das bedeutet aus der Sicht eines Formalismus, dass schon in der Definition einer unbeschränkt verzweigten Baumsprache Anzahlbedingungen formuliert werden müssen.

Um solche Einschränkungen zu realisieren, wurden verschiedene Ansätze für sogenannte *zählende Baumautomaten* entwickelt. Wir unterscheiden dazu in dieser Arbeit zwischen *lokalen* und *globalen* Anzahlbedingungen: Eine lokale Anzahlbedingung ist genau an die direkten Nachfolger eines bestimmten Knotens gerichtet, während eine globale Anzahlbedingung die Beschriftung eines ganzen Baumes betrifft. Es werden zwei zentrale Konzepte des Zählens benutzt: Die *Parikh-Abbildung* und die *Presburger Arithmetik*. Die Parikh-Abbildung nach Rohit J. Parikh bildet ein Wort über einem endlichen Alphabet auf einen Vektor von natürlichen Zahlen ab, in dem jeder Eintrag der Anzahl des Auftretens eines bestimmten Symbols innerhalb des Wortes entspricht [25]. Anzahlbedingungen werden durch *semi-lineare* Mengen formuliert, die wir in diesem Kontext *Constraint-Mengen* nennen. Eine formale Definition dieser Mengen werden wir später geben, ein einfaches Beispiel ist:

$$\left\{ \binom{m}{n} \mid m < n, m, n \in \mathbb{N} \right\} .$$

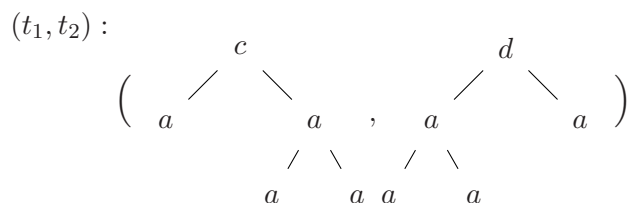
Der resultierende Vektor einer Parikh-Abbildung muss in einer solchen Menge enthalten sein, damit die Anzahlbedingung erfüllt ist.

Presburger Arithmetik, benannt nach Mojżesz Presburger, bezeichnet die Logik erster Stufe über den natürlichen Zahlen mit Addition. Presburger-Formeln definieren semi-lineare Mengen, und ihre Entscheidbarkeit wurde 1930 von Presburger [26] und unabhängig davon 1931 von Skolem gezeigt [36]. Anzahlbedingungen über Wörtern können durch Presburger-Formeln definiert werden, die für jedes Symbol des Alphabetes eine freie Variable haben.

Klaedtke und Rueß stellten 2002 einen Baumautomaten vor, der mit Hilfe einer erweiterten Parikh-Abbildung eine globale Anzahlbedingung an Eingabebäume stellen kann [20]. Muscholl, Seidl und Schwentick entwickelten 2003 ein Automatenmodell, das Presburger Logik nutzt, um lokale Anzahlbedingungen zu stellen [33, 34, 35]. Groz definierte 2006 sowohl einen Automaten, der die Parikh-Abbildung für die Definition lokaler Bedingungen nutzt, als auch einen Automaten, der eine globale Presburger-Bedingung an den ganzen Eingabebaum stellt [16]. Diese Modelle werden hier in verschiedenen Variationen vorgestellt und untersucht. Unter Beachtung der vorliegenden Resultate wird dann eine Hierarchie der *zählenden Baumsprachen* angegeben.

Es liegt nahe, den Begriff der rationalen Relation auf Bäume zu erweitern. Es existieren mehrere Definitionen für diese Erweiterung, da aber unterschiedliche Klassen definiert werden, ist die Definition der *rationalen Baumrelationen* nicht eindeutig festgelegt. Eine Beispielrelation, die durch die nachfolgenden Ansätze definierbar ist, wird die Anforderungen an einen Formalismus klarmachen.

**Beispiel 1.0.1.** Sei  $\mathcal{T}_1$  die Menge aller Bäume  $t_1$ , die die folgenden Bedingungen erfüllen: Die Wurzel ist mit  $c$  beschriftet und hat genau zwei Nachfolgeknoten, die mit  $a$  beschriftet sind. Alle mit  $a$  beschrifteten Knoten haben keinen oder genau zwei Nachfolgeknoten, die wiederum mit  $a$  beschriftet sind. Sei  $\mathcal{T}_2$  die Menge aller Bäume  $t_2$ , deren Wurzel mit  $d$  beschriftet ist und die ansonsten den Bäumen aus  $\mathcal{T}_1$  entsprechen. Eine Baumrelation ist eine Teilmenge des kartesischen Produktes von Baumsprachen:  $R \subseteq \mathcal{T}_1 \times \mathcal{T}_2$ . Die Relation  $R$  enthalte hier alle Paare von Bäumen  $(t_1, t_2)$ , sodass  $t_1$  und  $t_2$  gleich viele Knoten haben. Die folgende Abbildung zeigt ein solches 2-Tupel:



Das Beispiel macht deutlich, dass eine Synchronisation zwischen den einzelnen Tupel-elementen stattfinden muss.

Arnold und Dauchet erweiterten 1982 den Ansatz für rationale Wortrelationen von Nivat [23] durch die Definition von *Baumbimorphismen* [1]. Dabei wird ein Baum mit zwei Abbildungen auf ein 2-Tupel von Bäumen abgebildet. Eine reguläre Baumsprache und die beiden Abbildungen definieren gemeinsam eine 2-stellige Baumrelation.

Eine *Multivariable* ist eine Sequenz von paarweise verschiedenen Variablen. Diese Variablen beschriften Blätter von Bäumen und dienen als Platzhalter für Bäume; für die Variablen einer Multivariable müssen dabei bestimmte Operationen, zum Beispiel die Ersetzung durch einen Teilbaum, synchron durchgeführt werden. Raoult entwickelte 1992 die *Multivariablengrammatiken* [30]. Der initiale Ableitungsschritt dieser Baumgrammatiken erzeugt dabei ein Tupel von Bäumen, deren Blätter mit Variablen einer Multivariablen beschriftet sein können. Regeln zum Ersetzen von Variablen, die hier als Nichtterminalsymbole dienen, müssen dabei die gleichzeitige Ersetzung aller Variablen einer Multivariable gewährleisten. Raoult verglich seinen Ansatz mit dem von Arnold und Dauchet. Die durch Multivariablengrammatiken erzeugten Baumrelationen sind im Gegensatz zu denen, die durch einen Bimorphismus dargestellt werden können, nicht unter Konkatenation abgeschlossen. Er griff 1997 seinen Ansatz erneut auf und stellte rationale Baumrelationen äquivalent zu den Grammatiken durch *rationale Ausdrücke mit Multivariablen* dar [31]. Die Ausdrücke sind induktiv durch Abschluss unter Konkatenation und Kleene-Stern jeweils bezüglich einer Multivariablen definiert.

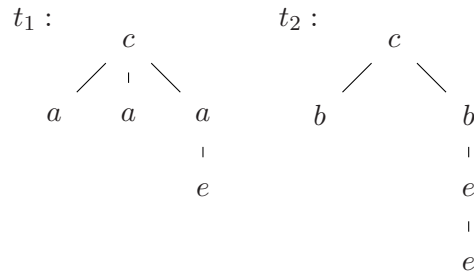
Radmacher übernahm 2007 diese Definition und entwickelte ein äquivalentes Automatenmodell, die *asynchronen Baumautomaten* [28, 29]. Die grundlegende Idee ist das Zusammenfassen von Zuständen zu Zustandstupeln, die analog zu den Multivariablen nur gemeinsam erreicht und verlassen werden können.

Ein zentrales Thema dieser Arbeit sind die rationalen unbeschränkt verzweigten Baumrelationen, für die bisher keine Definition existierte. Wir geben zunächst ein einfaches Beispiel an und betrachten dazu zwei Sprachen von unbeschränkt verzweigten Bäumen und eine entsprechende Relation.

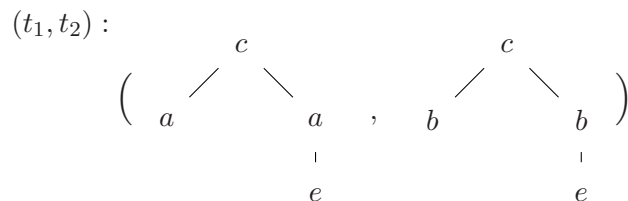
**Beispiel 1.0.2.** Sei  $\mathcal{T}_1$  die Menge aller Bäume  $t_1$ , die die folgenden Bedingungen erfüllen: Die Wurzel ist mit  $c$  beschriftet und hat beliebig viele Nachfolgeknoten. Alle diese Knoten sind mit  $a$  beschriftet. Der rechteste Nachfolger der Wurzel hat keinen oder genau einen Nachfolger, der mit  $e$  beschriftet ist. Alle  $e$ -Knoten haben keinen oder genau einen Nachfolger, der wiederum mit  $e$  beschriftet ist. Die Bäume der Sprache  $\mathcal{T}_2$  unterscheiden sich von denen der Sprache  $\mathcal{T}_1$  nur dahingehend, dass die direkten Nachfolger der Wurzel mit  $b$  beschriftet sind.

Die folgende Abbildung zeigt zwei Bäume  $t_1$  und  $t_2$ . Der erste Baum ist aus der Spra-

che  $\mathcal{T}_1$ , der zweite aus  $\mathcal{T}_2$ .



Wir setzen diese Baumsprachen nun in Beziehung zueinander. Die Menge  $R \subseteq \mathcal{T}_1 \times \mathcal{T}_2$  enthalte genau die Paare  $(t_1, t_2)$  von Bäumen  $t_1$  aus  $\mathcal{T}_1$  und  $t_2$  aus  $\mathcal{T}_2$  für die gilt, dass es für jeden mit  $a$  beschrifteten Knoten aus  $t_1$  genau einen mit  $b$  beschrifteten Knoten in  $t_2$  gibt; die gleiche Beziehung gelte für mit  $e$  beschriftete Knoten. Die folgende Abbildung zeigt ein Tupel aus dieser Relation:



Radmacher hat durch eine Erweiterung des asynchronen Baumautomaten auf unbeschränkt verzweigte Bäume einen Automaten für diese Art von Relationen konstruiert [28]. In dieser Arbeit stellen wir eine direkte Beziehung zwischen den rationalen Wortrelationen und den von diesem Automatenmodell erkannten Baumrelationen her, indem wir Wörter als *unäre Bäume*, in denen jeder Knoten genau einen oder keinen Nachfolger hat, auffassen. Da für diesen Sonderfall die Äquivalenz der Modelle gezeigt werden kann, definieren wir die unbeschränkt verzweigten Baumrelationen durch dieses Automatenmodell. In der weiteren Analyse stellt sich heraus, dass alle wichtigen Entscheidbarkeitsresultate der rationalen Wortrelationen auf die Baumrelationen übertragbar sind: die Inklusions-, Äquivalenz-, Universalitäts- und Schnittprobleme sind unentscheidbar. Es werden Algorithmen präsentiert, die das Nichtleerheits- und das Unendlichkeitsproblem entscheiden. Eine wichtige Eigenschaft der rationalen Wortrelationen ist, dass eine einstellige Relation genau eine reguläre Wortsprache definiert, während es einstellige rationale Baumrelationen gibt, die nicht-reguläre Mengen von Bäumen enthalten. Dies ist

als Schwäche des Modells anzusehen.

Wir führen in dieser Arbeit ein neues Automatenmodell ein, das genau die rationalen unbeschränkt verzweigten Baumrelationen erkennt, den *transitions-synchronisierten Baumautomaten*. Diese Automaten arbeiten auf Tupeln von unbeschränkt verzweigten Bäumen und erzwingen das gleichzeitige Ausführen bestimmter Transitionen, um eine Synchronisation zwischen Bäumen des Eingabetupels zu gewährleisten. Die Äquivalenz zu den asynchronen unbeschränkt verzweigten Baumautomaten wird gezeigt, und damit ist auch dieses Modell eine natürliche Erweiterung der rationalen Wortrelationen. Alle Entscheidbarkeitsresultate sind direkt übertragbar; zusätzlich geben wir einen Algorithmus an, der das Nichtleerheitsproblem entscheidet.

In der Folge schränken wir dieses Automatenmodell ein und erhalten die *transitions-separierten Baumautomaten*, die eine neue Klasse von Baumrelationen definieren. Es wird gezeigt, dass die einstelligen transitions-separierten Baumrelationen genau die regulären Baumsprachen definieren.

Wir wollen Anzahlbedingungen an Relationen von unbeschränkt verzweigten Bäumen stellen. Dazu kombinieren wir die transitions-synchronisierten Automaten mit den vorher eingeführten Ansätzen für zählende Baumautomaten. Wir erhalten somit Automaten, die auf Tupeln von unbeschränkt verzweigten Bäumen arbeiten und an die einzelnen Tupelelemente sowohl lokale als auch globale Anzahlbedingungen stellen können. Wir vergleichen die Modelle bezüglich ihrer Aussagekraft und erhalten eine nicht-lineare Hierarchie für die resultierenden Klassen von *zählenden Baumrelationen*. Es stellt sich heraus, dass alle entscheidbaren Eigenschaften der rationalen Wortrelationen bzw. der rationalen Baumrelationen auch für die zählenden Baumrelationen entscheidbar bleiben.

## **Gliederung der Arbeit**

Diese Arbeit ist wie folgt aufgebaut: In Kapitel 2 werden zunächst einige Grundlagen der Automatentheorie und der formalen Sprachen eingeführt. Es wird ein kurzer Überblick über rationale Wortrelationen und über Bäume bzw. Baumautomaten gegeben. Kapitel 3 beinhaltet eine ausführliche Einführung zu den zählenden Baumautomaten und zu grundlegenden Konzepten. Kapitel 4 behandelt verschiedene Konzepte für rationale Baumrelationen; es werden die *rationalen Ausdrücke mit Multivariablen* und die *asynchronen Baumautomaten* vorgestellt. In Kapitel 5 werden die rationalen Relationen über unbeschränkt verzweigten Bäumen durch *asynchrone unbeschränkt verzweigte Baumautomaten* definiert. Es werden verschiedene Eigenschaften betrachtet, und die Entscheidbarkeit oder Unentscheidbarkeit aller wichtigen Probleme wird gezeigt. Anschließend wird ein neues Automatenmodell entwickelt, das äquivalent zu dem des asynchronen

Baumautomaten ist. Dieses Modell wird eingeschränkt, und es resultieren die transitions-separierten Baumrelationen. In Kapitel 6 werden die Konzepte aus den Kapiteln 3 und 5 zusammengeführt und damit die Klasse der zählenden Baumrelationen definiert. Entscheidungsprobleme werden betrachtet, und die verschiedenen Automatenmodelle werden hinsichtlich ihrer Ausdrucksstärke miteinander verglichen. In Kapitel 7 werden schließlich eine kurze Zusammenfassung und ein Ausblick auf mögliche weitere Forschung gegeben.

### Danksagungen

Ich möchte hiermit allen danken, die in irgendeiner Form zu dieser Diplomarbeit beigetragen haben.

Zunächst danke ich Herrn Prof. Dr. Dr.h.c. Wolfgang Thomas für die Bereitstellung des Themas, die hervorragende Betreuung und die ständige Motivation, sowie Herrn Prof. Dr. Ir. Joost-Pieter Katoen dafür, dass er sich als Zweitgutachter dieser Arbeit zur Verfügung gestellt hat.

Ich danke Frank Radmacher und Wong Karianto, die ständig Probleme mit mir erörterten und für sorgfältiges Korrekturlesen zur Verfügung standen, sowie Frau Prof. Dr. Erika Ábrahám, Dennis Komm, Michael Holtmann, Roman Rabinovich, Bernd Puchala, Jörg Olschewski und Alex Spelten für Korrekturlesen, Ratschläge und Diskussionen.

Bedanken möchte ich mich auch bei allen weiteren Angestellten des *Lehrstuhls für Informatik 7* für eine sehr schöne Atmosphäre.

Mein besonderer Dank gilt meinen Eltern, die mein Studium erst ermöglicht haben.



# Kapitel 2

## Grundlagen

In diesem Kapitel werden verschiedene Begriffe und Konzepte eingeführt, die in dieser Arbeit benötigt werden. Zunächst wird ein kurzer Überblick über grundlegende Begriffe aus dem Bereich der Automatentheorie und der formalen Sprachen gegeben. Es folgt eine kurze, formale Einführung zu Bäumen und Baumautomaten. Zuletzt werden die *rationalen Relationen* über Wörtern formal definiert und kurz betrachtet.

### 2.1 Automaten und formale Sprachen

Ein *Alphabet*  $\Sigma$  ist eine endliche Menge von Symbolen.  $\Sigma^*$  bezeichnet die Menge aller endlichen Sequenzen  $w = a_1 \dots a_n$  über dem Alphabet, die *Menge der Wörter* über  $\Sigma$ , wobei  $n$  eine natürliche Zahl aus der Menge  $\mathbb{N}$  ist.  $\mathbb{N}^+$  ist die Menge der positiven natürlichen Zahlen. Die *Länge*  $|w| = n$  eines Wortes ist die Anzahl der Symbole in dem Wort.  $|w|_a$  sagt aus, wie oft Symbol  $a$  in Wort  $w$  auftritt. Das leere Wort  $\varepsilon$  hat die Länge 0. Die *Menge der nicht leeren Wörter* über  $\Sigma$  ist mit  $\Sigma^+$  bezeichnet. Die *Konkatenation* zweier Wörter  $v = v_1 \dots v_n$  und  $w = w_1 \dots w_m$  aus  $\Sigma^*$  mit  $n, m \in \mathbb{N}$  ist definiert durch  $v \cdot w = v_1 \dots v_n w_1 \dots w_m$ . Auf den Punkt wird dabei oft verzichtet:  $v \cdot w = vw$ . Das *kartesische Produkt* zweier Mengen  $\Sigma_1$  und  $\Sigma_2$  ist eine Menge von 2-Tupeln:  $\Sigma_1 \times \Sigma_2 = \{(a, b) \mid a \in \Sigma_1, b \in \Sigma_2\}$ . Ein *Präfix* eines Wortes  $w = w_1 \dots w_n$  ist ein Wort  $v = w_1 \dots w_m$  mit  $m \leq n$ .

Eine Sprache  $L$  ist eine Menge von Wörtern mit  $L \subseteq \Sigma^*$ . Die *Konkatenation zweier Sprachen*  $L$  und  $K$  ist definiert durch  $L \cdot K = \{v \cdot w \mid v \in L, w \in K\}$ . Die *iterierte Konkatenation* einer Sprache  $L$  ist definiert durch:  $L^0 := \{\varepsilon\}, L^n := L^{n-1} \cdot L$ . Der *Kleene-Abschluss* oder *Kleene-Stern* für eine Sprache  $L$  bezeichnet  $L^* = \bigcup_{n \in \mathbb{N}} L^n$ .

Mit diesen Begriffen werden die *regulären Ausdrücke* über einem Alphabet  $\Sigma$  definiert. Sie sind induktiv durch die atomaren regulären Ausdrücke  $\emptyset, \varepsilon, a \in \Sigma$  und ihre Abgeschlossenheit unter Konkatenation, Vereinigung und Kleene-Stern definiert. Die Menge der regulären Ausdrücke über  $\Sigma$  bezeichnen wir mit  $Reg(\Sigma)$ . Die Sprache eines regulären

Ausdrucks  $\mathcal{L}(r)$  bezeichnet die Menge der Wörter, die mit diesem Ausdruck erzeugt werden können. Die regulären Ausdrücke definieren genau die *regulären Sprachen*. Wir werden in dieser Arbeit aus Gründen der Lesbarkeit des öfteren reguläre Ausdrücke direkt mit Symbolen  $L_1, L_2 \dots$  bezeichnen und für ein Wort  $w$ , das mit einem Ausdruck  $L$  erzeugt werden kann,  $w \in L$  schreiben.

Ein *nichtdeterministischer endlicher Automat (NEA)* ist ein Tupel  $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$  mit einer endlichen Zustandsmenge  $Q$ , einem endlichen Alphabet  $\Sigma$ , einer Transitionsrelation  $\Delta \subseteq Q \times \Sigma \times Q$ , einem Startzustand  $q_0 \in Q$  und einer Endzustandsmenge  $F \subseteq Q$ . Ein Lauf des Automaten  $\mathcal{A}$  auf einem Wort  $w = a_1 \dots a_n$  ist eine Sequenz von Zuständen  $\rho = p_1 \dots p_n$  wobei  $p_1 = q_0$  und  $(p_i, a_{i+1}, p_{i+1}) \in \Delta, 1 \leq i \leq n$ . Ein Lauf heißt *akzeptierend*, wenn  $p_n \in F$ . Die Sprache eines Automaten ist definiert durch  $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \text{ hat einen akzeptierenden Lauf auf } w\}$ .

NEAs erkennen genau reguläre Sprachen. Sie sind unter Komplement und durch die *Produktautomatenkonstruktion* unter Schnitt und Vereinigung abgeschlossen.

Wir werden das folgende Lemma in späteren Kapiteln nutzen und referenzieren.

**Lemma 2.1.1.** *Sei  $\Sigma$  ein endliches Alphabet. Es ist entscheidbar, ob es ein Wort  $w$  über einem Alphabet  $\Sigma_1 \subseteq \Sigma$  für einen regulären Ausdruck  $\psi \in \text{Reg}(\Sigma)$  gibt, sodass  $w \in \mathcal{L}(\psi)$ .*

*Beweis.* Konstruiere zu dem regulären Ausdruck  $\psi$  einen äquivalenten, nichtdeterministischen endlichen Automaten  $\mathcal{A}_\psi$  und einen Automaten  $\mathcal{A}_{univ}$ , der genau  $\Sigma_1^*$  erkennt. Bilde den Produktautomaten  $\mathcal{A}_\psi \cap \mathcal{A}_{univ}$ . Die Sprache dieses Automaten ist genau die Sprache  $\mathcal{L}(\psi)$  eingeschränkt auf das Alphabet  $\Sigma_1$ . Da das Nichtleerheitsproblem für nichtdeterministische endliche Automaten entscheidbar ist, ist damit der Satz 2.1.1 gezeigt.  $\square$

Für ausführliche Einführungen in die Gebiete der Theorie der formalen Sprachen und Automaten sei auf das Standardwerk von Hopcroft und Ullman verwiesen [17]. Wir setzen grundlegende Kenntnisse in der mathematischen Logik voraus, zur Einführung empfiehlt sich das Buch von Thomas, Flum und Ebbinghaus [9].

## 2.2 Wortrelationen

Eine  $n$ -stellige *Relation*  $R$  auf Wörtern ist eine Teilmenge des kartesischen Produktes von Wortsprachen:

$$R \subseteq \Sigma_1 \times \dots \times \Sigma_n .$$

Die Elemente einer solchen Relation sind  $n$ -Tupel  $\bar{w} = (w_1, \dots, w_n)$  von Wörtern mit  $w_i \in \Sigma_i^*, 1 \leq i \leq n$ . Die *komponentenweise Konkatenation* zweier  $n$ -Tupel  $\bar{u} = (u_1, \dots, u_n)$

und  $\bar{v} = (v_1, \dots, v_n)$  ist definiert durch  $\bar{u} \cdot \bar{v} := (u_1v_1, \dots, u_nv_n)$ . Damit ist die komponentenweise Konkatenation zweier Relationen:

$$R \cdot S := \{\bar{u} \cdot \bar{v} \mid \bar{u} \in R, \bar{v} \in S\} .$$

Die *iterierte Konkatenation* von Relationen ist analog zu Sprachen gegeben durch  $R^0 := \{(\varepsilon, \dots, \varepsilon)\}$ ,  $R^n := R \cdot R^{n-1}$ ,  $n \in \mathbb{N}$ . Der *Kleene-Stern* ist definiert durch  $R^* := \bigcup_{n \in \mathbb{N}} R^n$ . Mit diesen Formalismen können die *rationalen Wortrelationen* induktiv definiert werden:

**Definition 2.2.1.** Für endliche Alphabete  $\Sigma_1, \dots, \Sigma_n$  gilt:

- $\emptyset$  ist rationale Relation
- $\bar{w} = (w_1, \dots, w_n) \in \Sigma_1^* \times \dots \times \Sigma_n^*$  ist rationale Relation
- Wenn die Relationen  $R_1$  und  $R_2$  rational sind, dann sind auch  $R_1 \cup R_2$ ,  $R_1 \cdot R_2$  und  $R_1^*$  rational.

Wir rufen uns erneut das Beispiel aus Kapitel 1 in Erinnerung:

$$R = \{(u, vu) \mid u, v \in \Sigma^*\} \subseteq \Sigma^* \times \Sigma^* .$$

Diese rationale Relation  $R$  enthält alle 2-Tupel von Wörtern, in denen das erste Wort ein Suffix des zweiten Wortes ist.  $R$  kann aufgrund der induktiven Definition rationaler Relationen durch den folgenden *rationalen Ausdruck* definiert werden:

$$\left( \left( \begin{smallmatrix} \varepsilon \\ a \end{smallmatrix} \right) + \left( \begin{smallmatrix} \varepsilon \\ b \end{smallmatrix} \right) \right)^* \cdot \left( \left( \begin{smallmatrix} a \\ a \end{smallmatrix} \right) + \left( \begin{smallmatrix} b \\ b \end{smallmatrix} \right) \right)^* .$$

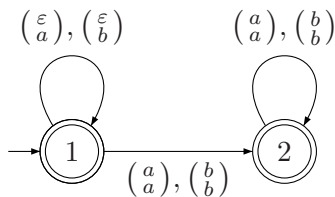
Äquivalent zu den rationalen Ausdrücken werden rationale Relationen von *asynchronen Automaten* erkannt.

**Definition 2.2.2.** Ein nichtdeterministischer asynchroner Automat ist ein Tupel  $\mathcal{A} = (Q, \Sigma_1, \dots, \Sigma_n, q_0, \Delta, F)$  mit einer endlichen Zustandsmenge  $Q$ , Alphabeten  $\Sigma_1, \dots, \Sigma_n$ , einem Startzustand  $q_0 \in Q$ , einer Endzustandsmenge  $F \subseteq Q$  und einer Transitionsrelation

$$\Delta \subseteq Q \times \Sigma_1^* \times \dots \times \Sigma_n^* \times Q .$$

Der Automat  $\mathcal{A}$  akzeptiert ein Tupel  $(u_1, \dots, u_n) \in \Sigma_1 \times \dots \times \Sigma_n$  wenn es auf  $\mathcal{A}$  einen Pfad von  $q_0$  aus zu einem Zustand aus  $F$  gibt mit der Beschriftung  $(u_1, \dots, u_n)$ . Die Relation, die von einem Automaten erkannt wird, ist gegeben durch

$$\mathcal{R}(\mathcal{A}) := \{(u_1, \dots, u_n) \in \Sigma_1^* \times \dots \times \Sigma_n^* \mid \mathcal{A} \text{ akzeptiert } (u_1, \dots, u_n)\} .$$

Abbildung 2.1: Asynchroner Automat  $\mathcal{A}$ 

In Abbildung 2.1 ist ein solcher Automat  $\mathcal{A}$  dargestellt. Er erkennt genau die Relation  $R$  des obigen Beispiels.

Es kann per Induktion über den Aufbau der rationalen Relationen gezeigt werden, dass sie genau durch die asynchronen Automaten definiert sind:

**Satz 2.2.1.** *Eine Relation  $R$  ist rational genau dann, wenn sie von einem asynchronen Automaten erkannt wird.*

Die Beweise der wichtigsten Entscheidbarkeitsresultate für rationale Relationen sind in [4] zu finden. Wir benötigen in dieser Arbeit die folgenden Unentscheidbarkeiten:

**Satz 2.2.2.** *Für rationale Relationen auf Wörtern sind Inklusion, Äquivalenz, Schnitt und Universalität unentscheidbar.*

## 2.3 Bäume und Baumautomaten

Wir geben in diesem Kapitel die Definitionen von beschränkt verzweigten und unbeschränkt verzweigten Baumsprachen sowie entsprechende Baumautomaten an.

Ein *Rangalphabet*  $\Sigma = \Sigma_0 \cup \dots \cup \Sigma_m$  mit  $m \in \mathbb{N}$  ist ein Alphabet, wobei jedem Symbol ein *Rang* zugewiesen wird.  $\Sigma_i$  für  $1 \leq i \leq m$  bedeutet in diesem Kontext, dass die Symbole dieser Teilmenge des gesamten Alphabetes den Rang  $i$  haben.

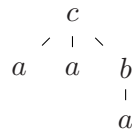
**Definition 2.3.1.** Ein *beschränkt verzweigter Baum* über einem Rangalphabet  $\Sigma$  ist ein Paar  $t = (dom_t, val_t)$  mit einer endlichen Menge  $dom_t \subseteq \mathbb{N}^*$ , der Menge von Knoten des Baumes, sodass:

- $dom_t$  ist unter Vorgängern abgeschlossen:  $ui \in dom_t \Rightarrow u \in dom_t$  für  $u \in \mathbb{N}^*$ ,  $i \in \mathbb{N}$
- $dom_t$  ist unter linken Geschwisterknoten abgeschlossen:  $ui \in dom_t \Rightarrow uj \in dom_t$  für  $u \in \mathbb{N}^*$ ,  $j \leq i$

- Die Funktion  $val_t : dom_t \rightarrow \Sigma$  beschriftet einen Knoten  $u \in dom_t$ , der genau  $i$  Nachfolger hat, mit einem Symbol  $a^i \in \Sigma_i$ .

In einem beschränkt verzweigten Baum bestimmt der Rang des Symbols, mit dem ein Knoten beschriftet ist, genau die Anzahl seiner Nachfolger. Der Knoten  $u \in dom_t$  mit  $u = \varepsilon$  ist die *Wurzel* des Baumes, sie hat keine Vorgänger. Die Knoten, die durch  $val_t$  mit Symbolen des Rangs 0 beschriftet werden, sind die *Blätter* des Baumes, sie haben keine Nachfolger. Die Höhe eines Baumes ist definiert durch  $\max\{|u| \mid u \in dom_t\}$ .

Die Menge aller Bäume über einem Alphabet  $\Sigma$  bezeichnen wir mit  $\mathcal{T}_\Sigma$ . Eine *Baum-sprache*  $\mathcal{T} \subseteq \mathcal{T}_\Sigma$  ist eine Menge von Bäumen. Die unten stehende Abbildung zeigt einen beschränkt verzweigten Baum über dem Rangalphabet  $\Sigma = \{c^3, b^1, a^0\}$ .



Eine andere Schreibweise für diesen Baum ist  $c(a, a, b(a))$ . Da die Anzahl der Nachfolger eines Knotens durch den Rang eindeutig festgelegt ist, kann hier sogar auf die Klammerung verzichtet werden:  $caaba$ .

Wir definieren nun einen bottom-up Baumautomaten. Ein bottom-up Automat liest zunächst die Blätter eines Baumes ein und erreicht zuletzt die Wurzel.

**Definition 2.3.2.** Ein nichtdeterministischer beschränkt verzweigter Baumautomat ist ein Tupel  $\mathcal{A} = (Q, \Sigma, \Delta, F)$  mit einer endlichen Zustandsmenge  $Q$ , einem Rangalphabet  $\Sigma = \Sigma_0 \cup \dots \cup \Sigma_m$ , einer Endzustandsmenge  $F$  und einer Transitionsrelation  $\Delta$  mit

$$\Delta \subseteq \bigcup_{i=0}^m (Q^i \times \Sigma_i \times Q) .$$

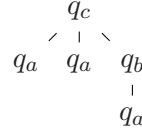
Ein Lauf des Automaten auf einem Baum  $t$  ist eine Abbildung  $\rho : dom_t \rightarrow Q$ , sodass alle Knoten des Baumes entsprechend ihrer Beschriftungen mit geeigneten Transitionen auf einen Zustand aus  $Q$  abgebildet werden. Ein Baum wird akzeptiert, wenn  $\rho(\varepsilon) \in F$ . Die Sprache eines Baumautomaten ist definiert durch  $\mathcal{T}(\mathcal{A}) = \{t \in \mathcal{T}_\Sigma \mid \mathcal{A} \text{ akzeptiert } t\}$ .

Wir stellen in dieser Arbeit den Lauf eines Automaten auf einem Baum grundsätzlich durch einen gleich strukturierten Baum dar, der mit Zuständen des Automaten beschriftet ist.

**Beispiel 2.3.1.** Sei ein nichtdeterministischer beschränkt verzweigter Baumautomat gegeben durch  $\mathcal{A} = (\{q_a, q_b, q_c\}, \{c^3, b^1, a^0\}, \Delta, \{q_c\})$  mit

$$\Delta = \{(a, q_a), (q_a, b, q_c), (q_a q_a q_c, c, q_c)\} .$$

Die unten stehende Abbildung zeigt einen akzeptierenden Lauf des Automaten  $\mathcal{A}$  auf dem oben dargestellten beschränkt verzweigten Baum.



Die nichtdeterministischen Baumautomaten definieren genau die *regulären Baumsprachen*.

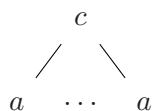
*Unbeschränkt verzweigte Bäume* sind nicht über einem Rangalphabet definiert. Die Anzahl der Nachfolger eines Knotens ist nicht durch seine Beschriftung bestimmt. Man kann somit beliebige Verzweigungsgrade definieren, allerdings hat ein Knoten grundsätzlich nur endlich viele Nachfolger. Ein unbeschränkt verzweigter Baum ist daher endlich.

Ein *Beschriftungsalphabet* ist eine nicht leere, endliche Menge von Symbolen.

**Definition 2.3.3.** Ein *unbeschränkt verzweigter Baum* über einem Beschriftungsalphabet  $\Sigma$  ist ein Paar  $t = (dom_t, val_t)$  mit einer endlichen Menge  $dom_t \subseteq \mathbb{N}^*$ , sodass:

- $dom_t$  ist unter Vorgängern abgeschlossen:  $ui \in dom_t \Rightarrow u \in dom_t$  für  $u \in \mathbb{N}^*, i \in \mathbb{N}$
- $dom_t$  ist unter linken Geschwisterknoten abgeschlossen:  $ui \in dom_t \Rightarrow uj \in dom_t$  für  $u \in \mathbb{N}^*, j \leq i$
- Die Funktion  $val_t : dom_t \rightarrow \Sigma$  beschriftet einen Knoten  $u \in dom_t$  mit einem Symbol aus  $\Sigma$ .

Die Menge aller unbeschränkt verzweigten Bäume über einem Beschriftungsalphabet  $\Sigma$  ist bezeichnet mit  $\mathcal{T}_\Sigma$ . Eine Baumsprache  $\mathcal{T} \subseteq \mathcal{T}_\Sigma$  ist dann eine Sprache über unbeschränkt verzweigten Bäumen, wenn  $\Sigma$  ein Beschriftungsalphabet ist. *Reguläre unbeschränkt verzweigte Baumsprachen* setzen eine Regularität für die Kinder eines Knotens voraus, das heißt, die Beschriftungen der Nachfolger eines Knotens müssen von links nach rechts gelesen ein Wort einer regulären Sprache bilden. Unten stehend ist eine solche Sprache dargestellt, in der die mit  $c$  beschriftete Wurzel beliebig viele Nachfolger hat, die mit  $a$  beschriftet sind. Die Beschriftungen der Nachfolger sollen dem regulären Ausdruck  $a^*$  genügen.



Wir definieren nun ein Automatenmodell, das in seinen Transitionen eben diese Regularität nutzt und damit die regulären Baumsprachen definiert:

**Definition 2.3.4.** Ein nichtdeterministischer unbeschränkt verzweigter Baumautomat ist ein Tupel  $\mathcal{A} = (Q, \Sigma, \Delta, F)$  mit einer endlichen Zustandsmenge  $Q$ , einem Beschriftungsalphabet  $\Sigma$ , einer Endzustandsmenge  $F$  und einer Transitionsrelation  $\Delta$  mit

$$\Delta \subseteq \text{Reg}(Q) \times \Sigma \times Q .$$

Ein Lauf des Automaten auf einem Baum  $t$  ist eine Abbildung  $\rho : \text{dom}_t \rightarrow Q$ , sodass alle Knoten des Baums entsprechend ihrer Beschriftungen und mit geeigneten Transitionen auf einen Zustand aus  $Q$  abgebildet werden. Ein Baum wird akzeptiert, wenn  $\rho(\varepsilon) \in F$ . Die Sprache eines Baumautomaten ist definiert durch  $\mathcal{T}(\mathcal{A}) = \{t \in T_\Sigma \mid \mathcal{A} \text{ akzeptiert } t\}$ .

**Beispiel 2.3.2.** Sei ein unbeschränkt verzweigter Baumautomat gegeben durch  $\mathcal{A} = (\{q_a, q_c\}, \{a, c\}, \Delta, \{q_c\})$  mit  $\Delta = \{(a, q_a), (q_a^*, c, q_c)\}$ .

Dieser nichtdeterministische Automat erkennt genau die oben dargestellte unbeschränkt verzweigte Baumsprache. Ein deterministisches Modell ist in [8] aufgeführt.

Für weitere Ausführungen zu Bäumen und Baumautomaten sei auf [7] verwiesen.



# Kapitel 3

## Anzahlbedingungen

In diesem Kapitel werden verschiedene Baumautomaten eingeführt, die an die Beschriftungen von Bäumen Anzahlbedingungen stellen können. Solche Bedingungen können durch logische Formeln formuliert werden, welchen dann bestimmte Teile der Beschriftung eines Eingabebaumes genügen müssen. Ein anderes Konzept ist ein in seinen Transitionen zählender Automat.

Wir unterscheiden hier konkret zwischen *lokalen Anzahlbedingungen* und *globalen Anzahlbedingungen*. Eine lokale Anzahlbedingung gilt genau für die Kinder eines Knotens in einem Baum, eine globale Anzahlbedingung gilt für die gesamte Beschriftung. So kann man z.B. lokal fordern, dass unter den direkten Nachfolgern eines mit  $c$  beschrifteten Knotens mehr Kinder mit  $b$  als mit  $a$  beschriftet sind. Global könnte es im Gegensatz dazu mehr  $a$ -Beschriftungen als  $b$ -Beschriftungen geben. Es werden dazu die *Presburger Arithmetik* und *Parikh-Wortautomaten* vorgestellt, diese Konzepte werden mit den üblichen Baumautomaten für unbeschränkt verzweigte Bäume kombiniert bzw. entsprechend erweitert. Die Klasse der *zählenden Baumsprachen* bezeichnen wir mit `Count`.

### 3.1 Semi-lineare Mengen und Presburger-Arithmetik

Zunächst führen wir die Begriffe einer *linearen Menge* und einer *semi-linearen Menge*, die zuerst 1961 von Parikh [24] vorgestellt wurde, ein. Sei dazu

$$\{\bar{x} \mid \bar{x} = (x_1, \dots, x_n) \in \mathbb{N}^n, n \in \mathbb{N}\}$$

die Menge aller Vektoren von natürlichen Zahlen der Dimension  $n$ . Dabei sei  $\bar{x} + \bar{y}$  die komponentenweise Addition zweier Vektoren und  $t \cdot \bar{x}$  für  $t \in \mathbb{N}$  die komponentenweise Multiplikation aller Komponenten von  $\bar{x}$  mit der Konstanten  $t$ . Seien  $C$  und  $D$  Teilmengen von  $\mathbb{N}^n$  und  $t_1, \dots, t_m$  Konstanten aus  $\mathbb{N}$ . Dann ist  $\mathcal{L}(C, D)$  die Menge aller  $\bar{x} \in \mathbb{N}^n$  von der Form

$$\bar{x} = \bar{c} + t_1 \cdot \bar{d}_1 + \dots + t_m \cdot \bar{d}_m$$

mit  $\bar{c} \in C, \bar{d}_1, \dots, \bar{d}_m \in D$ . Die  $\bar{d}_1, \dots, \bar{d}_m$  bilden eine möglicherweise leere, endliche Folge von Vektoren aus  $D$ . Für  $C = \{c\}$  schreiben wir  $\mathcal{L} = (c, D)$ .

**Definition 3.1.1.** Eine Menge  $X \subseteq \mathbb{N}^n$  heißt *linear*, wenn ein  $c \in \mathbb{N}^n$  und eine endliche Teilmenge  $D$  von  $\mathbb{N}^n$  existieren, sodass  $X = \mathcal{L}(c, D)$ .

Eine Menge  $Y$  heißt *semi-linear*, wenn sie die endliche Vereinigung linearer Mengen ist.

Ginsburg und Spanier zeigen in [15] die Abschlusseigenschaften semi-linearer Mengen.

**Satz 3.1.1.** *Die Familie der semi-linearen Mengen aus  $\mathbb{N}^n$  ist abgeschlossen unter Vereinigung, Schnitt, Komplement und Projektion.*

**Beispiel 3.1.1.**

1. Sei die Menge  $X_1 = \{(x_1, x_2) \in \mathbb{N}^2 \mid 2x_1 < x_2\}$  gegeben.  $X_1$  ist linear, da sie durch

$$\left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix} + k_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + k_2 \begin{pmatrix} 1 \\ 2 \end{pmatrix} \mid k_1, k_2 \in \mathbb{N} \right\}$$

definiert ist.

2. Sei  $X_2 = \{(x_1, x_2) \in \mathbb{N}^2 \mid x_1 \text{ ist eine gerade Zahl}\}$ .  $X_2$  ist durch

$$\left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix} + k_1 \begin{pmatrix} 2 \\ 0 \end{pmatrix} + k_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mid k_1, k_2 \in \mathbb{N} \right\}$$

definierbar.

3. Die Menge  $X_3 = X_1 \cup X_2$  ist laut Definition eine semi-lineare Menge. Sie ist sogar linear, da sie durch

$$\left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix} + k_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + k_2 \begin{pmatrix} 2 \\ 4 \end{pmatrix} \mid k_1, k_2 \in \mathbb{N} \right\}$$

definiert ist.

Die *Presburger-Arithmetik*, benannt nach Mojżesz Presburger, ist die Logik erster Stufe der natürlichen Zahlen mit Addition, Ordnungsrelation und Gleichheit über der Struktur  $\xi = (\mathbb{N}, \leq, +)$ . Eine *Presburger-Formel*  $\varphi$  ist eine Formel erster Stufe über  $\xi$ . Wir schreiben  $\xi \models \varphi[a_1, \dots, a_n]$  für eine Formel  $\varphi(x_1, \dots, x_n)$  und  $a_1, \dots, a_n \in \mathbb{N}$ , wenn die freien Variablen  $x_1, \dots, x_n$  als natürliche Zahlen  $a_1, \dots, a_n$  interpretiert werden und  $\varphi$  dadurch erfüllt ist. Eine solche Interpretation liefert Presburger-Formeln ohne freie Variablen, diese heißen *Presburger-Terme*. Für eine Formel  $\varphi(x_1, \dots, x_n)$  definieren wir die Menge

$$\llbracket \varphi \rrbracket := \{(a_1, \dots, a_n) \in \mathbb{N}^n \mid \xi \models \varphi[a_1, \dots, a_n]\}.$$

Die Vektoren  $(a_1, \dots, a_n)$  bilden die *Menge der Interpretationen* von  $\varphi$ . Eine Menge  $A$  von Vektoren der Dimension  $n \geq 1$  heißt *Presburger-Menge*, wenn es eine Presburger-Formel  $\varphi$  mit  $n$  freien Variablen gibt, sodass  $A = \llbracket \varphi \rrbracket$  gilt.

Es folgt eine formale Definition nach Ginsburg und Spanier [15].

**Definition 3.1.2.** Die *Menge aller Presburger-Formeln*  $\mathcal{P}$  ist die kleinste Klasse von Formeln, für die gilt:

- Für  $t_i, t'_i \in \mathbb{N}, 0 \leq i \leq n$  ist

$$t_0 + \sum_1^n t_i x_i = t'_0 + \sum_1^n t'_i x_i$$

eine Presburger-Formel aus  $\mathcal{P}$ .

- Wenn  $\varphi_1$  und  $\varphi_2$  in  $\mathcal{P}$  sind, dann ist auch  $\varphi_1 \wedge \varphi_2$  in  $\mathcal{P}$ .
- Wenn  $\varphi_1$  und  $\varphi_2$  in  $\mathcal{P}$  sind, dann ist auch  $\varphi_1 \vee \varphi_2$  in  $\mathcal{P}$ .
- Wenn  $\varphi$  in  $\mathcal{P}$  ist, dann ist auch  $\neg\varphi$  in  $\mathcal{P}$ .
- Wenn  $\varphi(x_1, \dots, x_n)$  in  $\mathcal{P}$  ist, dann ist auch  $\exists x_i(\varphi(x_1, \dots, x_n))$  in  $\mathcal{P}$  für  $1 \leq i \leq n$ .

Wir benutzen Presburger-Formeln in dieser Arbeit wie oben erwähnt direkt mit einer Ordnungsrelation und einigen zusätzlichen Vereinfachungen. Die hier verwendeten Formeln werden von der folgenden Grammatik erzeugt:

$$\begin{aligned} \varphi ::= & \quad x \leq n \mid x = n \mid x + y = z \\ & \quad \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \\ & \quad \mid y = nx \mid \exists x(\varphi) \mid \forall x(\varphi) \mid \mathbf{true} \end{aligned}$$

Dabei sind die  $x, y, z$  Variablen, und  $n \in \mathbb{N}$  ist eine Konstante. Durch die induktive Definition der Presburger Arithmetik ist klar, dass diese unter Vereinigung, Schnitt, Komplement und Projektion abgeschlossen ist.

Bereits 1930 bzw. 1931 zeigten sowohl Presburger [26] als auch Skolem [36] die Entscheidbarkeit der Presburger-Arithmetik.

**Satz 3.1.2** (Presburger, Skolem). *Presburger Arithmetik ist entscheidbar.*

Presburger bewies diesen Satz durch die Entwicklung eines Algorithmus zur Eliminierung von Quantoren. Dabei wird zu einer gegebenen Presburger-Formel eine äquivalente,

quantorenfreie Formel berechnet. *Quantorenfreie Presburger-Formeln* sind die Booleschen Kombinationen atomarer Formeln. Sie werden für  $d \in \mathbb{N}$  von der Grammatik

$$\begin{aligned} \varphi & ::= t = 0 \mid t \equiv 0 \pmod{d} \\ & \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg \varphi \\ t & ::= 0 \mid 1 \mid +x \mid -x \mid t_1 + t_2 \end{aligned}$$

erzeugt. Eine solche Form ist leicht zu evaluieren, und dies hat gegenüber einem direkten Entscheidungsverfahren den Vorteil, dass nicht nur die Entscheidbarkeit überprüft wird, sondern implizit auch eine gültige Interpretation der nur aus freien Variablen bestehenden Formel angegeben wird, wenn sie erfüllbar ist. Presburger führte den Beweis über der Struktur  $(\mathbb{Z}, +, <)$ , während hier  $\xi = (\mathbb{N}, \leq, +)$  betrachtet wird. Dieser Unterschied kann vernachlässigt werden, wie Klädtke zeigte [19]. Rabin und Fischer gaben 1974 in [12] eine doppelt exponentielle, nichtdeterministische untere Schranke für die Elimination von Quantoren an, 1980 zeigte Berman, dass die genaue Komplexität  $\text{LINA-TIME } 2^{2^{O(n)}}$  ist [3].

Ein anderer Ansatz für die Evaluation von Presburger Logik ist die Entwicklung von Wortautomaten, die äquivalent zu einer Presburger-Formel konstruiert werden. Dabei werden Vektoren von natürlichen bzw. ganzen Zahlen eindeutig als Wörter kodiert; ein Automat akzeptiert dann genau die Wörter, durch deren äquivalente Vektoren die entsprechende Presburger-Formel wahr wird. Für Details sei beispielsweise auf die Arbeiten von Büchi [2], Wolper and Boigelot [39] und Klaedtke verwiesen, der die obere Schranke von  $2^{2^{O(n)}}$  für die Anzahl der Zustände eines minimalen deterministischen Automaten für eine Presburger-Formel zeigte [19].

Aus Gründen der Komplexität ist die *Existentielle Normalform* einer Presburger-Formel interessant. *Existentielle Presburger-Formeln* werden für  $d \in \mathbb{N}$  von der Grammatik

$$\begin{aligned} \varphi & ::= t = 0 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x(\varphi) \\ t & ::= 0 \mid 1 \mid +x \mid -x \mid t_1 + t_2 \end{aligned}$$

erzeugt. Eine Formel ist in existentieller Normalform, wenn sie existentiell und von der Form

$$\exists x_1, \dots, x_k \bigvee_{i=1}^m \left( \bigwedge_{j=1}^n (t_{ij}) \right)$$

ist. Jede Presburger-Formel kann in eine äquivalente quantorenfreie Formel transformiert werden [26]. Seidl, Schwentick und Muscholl zeigen in [34], dass durch einen weiteren

Schritt eine solche Formel in existentielle Normalform transformiert werden kann. Die Entscheidbarkeit für existentielle Presburger-Formeln ist  $NP$ -vollständig.

Ginsburg und Spanier zeigen in [15] die folgende Eigenschaft der durch Presburger-Formeln definierten Mengen:

**Satz 3.1.3** (Ginsburg, Spanier). *Die Klasse der Presburger Mengen über  $\mathbb{N}^n$  entspricht genau der Klasse der semi-linearen Mengen über  $\mathbb{N}^n$ . Die eine Charakterisierung einer Menge ist dabei jeweils effizient aus der anderen berechenbar.*

**Beispiel 3.1.2.** Die drei Mengen  $X_1, X_2, X_3$  aus Beispiel 3.1.1 sind durch Presburger-Formeln  $\varphi_1, \varphi_2, \varphi_3$  beschreibbar. Sei dazu

- $\varphi_1(x_1, x_2) := 2 \cdot x_1 < x_2$
- $\varphi_2(x_1, x_2) := \exists y(2 \cdot y = x_1)$
- $\varphi_3(x_1, x_2) := \exists y(2 \cdot x_1 = y \wedge 2 \cdot x_1 < x_2)$

## 3.2 Parikh-Zählen

In diesem Kapitel wird eine Einführung zu zählenden Wortautomaten, den *Parikh-Automaten* gegeben. Diese Automaten wurden 2002 auf der Grundlage des *Parikh-Bildes* [25] von Klaedtke und Rueß entwickelt [20], die hier aufgeführten Definitionen und Ergebnisse stammen aus dieser Veröffentlichung. Für eine detailliertere Auseinandersetzung mit diesem Thema sei neben der Originalarbeit auf die Diplomarbeit von Karianto [18] verwiesen, der in seiner Arbeit den Ansatz der endlichen Parikh-Wortautomaten zur Entwicklung von Parikh Pushdown Automaten nutzt.

Parikh-Automaten arbeiten auf endlichen Wörtern mit einer gegenüber den bekannten endlichen Wortautomaten erweiterten Akzeptierbedingung. Ein solcher Automat erreicht für Wörter einer bestimmten Sprache - der Sprache des Automaten - einen Endzustand mit Lesen des letzten Symbols eines Eingabewortes, zusätzlich muss eine gegebene Anzahlbedingung für die Symbole des Eingabealphabetes erfüllt sein. Man kann mit diesen Automaten also durch Anzahlbedingungen eingeschränkte Sprachen über Wörtern definieren.

### 3.2.1 Parikh-Abbildung

Durch die Parikh-Abbildung [25] werden über einem Alphabet definierte Wörter auf Vektoren abgebildet. Dabei ist jedes Symbol eines Alphabetes eindeutig einem Einheitsvektor über den natürlichen Zahlen zuzuordnen, die Dimension dieser Vektoren entspricht

dann der Kardinalität des Alphabets. Um eine solche Abbildung korrekt definieren zu können, muss eine lineare Ordnung auf dem Alphabet gefordert werden.

**Definition 3.2.1.** Sei  $\Sigma = \{a_1, \dots, a_n\}$  ein linear geordnetes Alphabet, und sei  $\bar{e}_i \in \mathbb{N}^n$  der Einheitsvektor für  $n \in \mathbb{N}$ , wobei die  $i$ -te Komponente den Wert 1 hat und alle anderen den Wert 0. Die Parikh-Abbildung  $\Phi : \Sigma^* \rightarrow \mathbb{N}^n$  ist definiert durch

- $\Phi(a_i) = \bar{e}_i$
- $\Phi(uv) = \Phi(u) + \Phi(v)$
- $\Phi(\varepsilon) = 0^n$ .

Vereinfacht definiert ist  $\Phi : \Sigma^* \rightarrow \mathbb{N}^n$  für  $w \in \Sigma^*$ :

$$\Phi(w) = (|w|_{a_1}, \dots, |w|_{a_n})$$

Intuitiv wird ein Wort auf einen Vektor abgebildet, der genau die Kardinalitäten aller Symbole aus  $\Sigma$  in  $w$  angibt. Das folgende Beispiel verdeutlicht die Anwendung einer solchen Abbildung.

**Beispiel 3.2.1.** Sei  $\Sigma = \{a, b, c, d\}$  und  $w = abaabca \in \Sigma^*$ . Dann ist

$$\Phi(w) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 1 \\ 0 \end{pmatrix} \in \mathbb{N}^4$$

Parikh zeigte, dass die folgende Eigenschaft gilt [25]:

**Satz 3.2.1** (Parikh). *Das Parikh-Bild einer kontextfreien Sprache ist eine semi-lineare Menge.*

### 3.2.2 Parikh-Automat

Das Ziel ist die Definition eines Automaten, der mit Hilfe des Parikh-Bildes Symbole in einem Wort zählen kann. Am Ende eines akzeptierenden Laufes des Automaten soll das Parikh-Bild des Eingabewortes zur Verfügung stehen; wenn dieser Vektor Element einer bestimmten Menge von Vektoren ist, akzeptiert der Automat das Eingabewort.

Um während eines Laufes zählen zu können, wird durch die Transitionen festgelegt, welche Aktion für ein bestimmtes Symbol des Eingabealphabetes durchgeführt wird. Dazu wird als Alphabet des Automaten nicht mehr das Alphabet  $\Sigma$ , über dem Eingabewörter gebildet sind, betrachtet, sondern das kartesische Produkt von  $\Sigma$  und einer

Menge von Vektoren  $D$  über den natürlichen Zahlen. Diese Vektoren müssen im Gegensatz zu denen des klassischen Parikh-Bildes keine Einheitsvektoren mehr sein. Zunächst erweitern wir das Parikh-Bild dementsprechend.

**Definition 3.2.2.** Seien  $\Sigma$  ein endliches Alphabet und  $D$  eine nicht-leere Teilmenge von  $\mathbb{N}^n$  für ein  $n \in \mathbb{N}$ . Die  $\Sigma$ -Projektion  $\Psi : (\Sigma \times D)^* \rightarrow \Sigma^*$  ist definiert durch

$$\Psi(a, \bar{d}) := a, \text{ für } (a, \bar{d}) \in \Sigma \times D, \text{ und } \Psi(uv) := \Psi(u)\Psi(v), \text{ für } u, v \in (\Sigma \times D)^*.$$

Die erweiterte Parikh-Abbildung  $\Phi : (\Sigma \times D)^* \rightarrow \mathbb{N}^n$  ist definiert durch

$$\Phi(a, \bar{d}) := \bar{d}, \text{ für } (a, \bar{d}) \in \Sigma \times D, \text{ und } \Phi(uv) := \Phi(u) + \Phi(v), \text{ für } u, v \in (\Sigma \times D)^*.$$

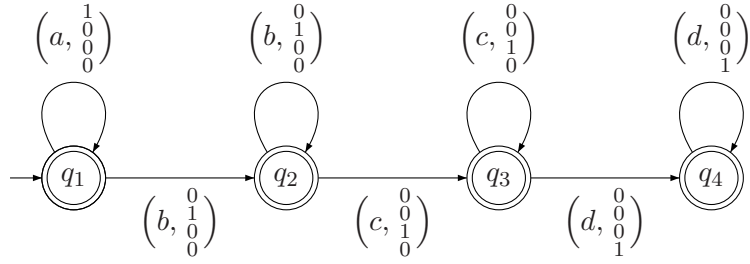
Mit dieser Definition ist es möglich, einem Symbol beliebige Vektoren aus der Menge  $D$  zuzuordnen. Wird einem Symbol  $a_i \in \Sigma$  eindeutig der Einheitsvektor  $\bar{e}_i \in \mathbb{N}^n$  mit  $n = |\Sigma|$  zugeordnet, entspricht das *erweiterte Parikh-Bild* dem klassischen Bild aus Definition 3.2.1. Das erweiterte Parikh-Bild liefert zu einem Wort einen Vektor. Durch die Definition einer Menge von Vektoren, in der ein resultierender Vektor enthalten sein muss, kann man eine reguläre Sprache  $L$  einschränken, indem man die  $\Sigma$ -Projektion dieser Sprache betrachtet. Eine solche *Constraint-Menge*  $C$  stellt implizit Anzahlbedingungen an Wörter einer Sprache.

**Definition 3.2.3.** Seien  $\Sigma$  ein Alphabet,  $D$  eine nicht-leere, endliche Teilmenge von  $\mathbb{N}^n$  für ein  $n \in \mathbb{N}$  und sei  $C \subseteq \mathbb{N}^n$  eine Constraint-Menge. Für eine Sprache  $L \subseteq (\Sigma \times D)^*$  ist die Einschränkung von  $L$  bezüglich  $C$  definiert durch

$$L \upharpoonright_C = \{\Psi(w) \mid w \in L, \Phi(w) \in C\}.$$

Analog zu Klaedtke und Rueß [20] werden hier zur Formulierung von Anzahlbedingungen ausschließlich semi-lineare Mengen  $C$  nach Definition 3.1.1 betrachtet. Die Akzeptierbedingung eines endlichen Automaten auf Wörtern wird zur Definition der Parikh-Automaten wie folgt erweitert. Ein Wort  $w$  über  $(\Sigma \times D)^*$  ist eine Folge von Tupeln  $(a, \bar{d}) \in \Sigma \times D$ . Ein solches Wort wird akzeptiert, wenn zum einen die Folge der Symbole  $a$  verknüpft mit beliebigen Vektoren  $d$  zum Erreichen eines Endzustandes führt, zum anderen die Summe aller Vektoren  $d$  ein Vektor aus der Menge  $C$  ist. In dieser Arbeit bezeichnen wir einen *endlichen Parikh-Automaten auf Wörtern* mit *Parikh-Automat*.

**Definition 3.2.4.** Ein Parikh-Automat der Dimension  $n \geq 1$  ist ein Paar  $(\mathcal{A}, C)$ , wobei  $\mathcal{A}$  ein endlicher Automat auf Wörtern und  $C$  eine semi-lineare Teilmenge von  $\mathbb{N}^n$  ist.  $\mathcal{A}$  ist über einem Alphabet der Form  $\Sigma \times D$  definiert, dabei ist  $\Sigma$  ein endliches Alphabet und  $D$  eine endliche, nicht-leere Teilmenge von  $\mathbb{N}^n$ .

Abbildung 3.1: Parikhautomat  $\mathcal{A}$ 

Wir nennen in diesem Kontext  $\mathcal{A}$  die *Automatenkomponente* und  $C$  die *Constraint-Menge*. Ein Parikh-Automat erkennt die Sprache  $\mathcal{L}(\mathcal{A}, C) := \mathcal{L}(\mathcal{A}) \upharpoonright_C$ . Um die Funktionsweise besser zu erläutern, geben wir ein Beispiel eines solchen Automaten an.

**Beispiel 3.2.2.** Sei der Parikh-Automat  $(\mathcal{A}, C)$  gegeben durch den Automaten  $\mathcal{A}$  aus Abbildung 3.1 und durch

$$C = \left\{ \begin{pmatrix} d \\ d \\ e \end{pmatrix} \mid d, e \in \mathbb{N}^+ \right\}.$$

Der Automat  $\mathcal{A}$  definiert unter Vernachlässigung der Constraint-Menge die Sprache  $L = a^*b^*a^*c^*$ . Die Einschränkung von  $L$  bezüglich  $C$  und damit die Sprache des Parikh-Automaten ist  $L \upharpoonright_C = L((\mathcal{A}, C)) = \{a^n b^n a^m c^m \mid n, m > 0\}$ .

Der Vollständigkeit halber geben wir noch die Definition eines *deterministischen Parikh-Automaten* an, der Determinismus zählender Automaten ist aber nicht das Thema dieser Arbeit.

**Definition 3.2.5.** Ein Parikh-Automat  $(\mathcal{A}, C)$  mit  $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$  ist deterministisch, wenn für jeden Zustand  $q \in Q$  und für jedes Tupel  $(a, \bar{d}) \in \Sigma \times D$  gilt, dass

- $|\delta(q, (a, \bar{d}))| \leq 1$
- Wenn  $|\delta(q, (a, \bar{d}))| = 1$ , dann  $|\delta(q, (a, \bar{d}'))| = 0$  für alle  $\bar{d}' \neq \bar{d}$

Parikh-Automaten sind abgeschlossen unter Vereinigung, Schnitt und Konkatenation, aber nicht unter Komplement.

**Satz 3.2.2.** *Das Nichtleerheitsproblem für Parikh-Automaten ist entscheidbar.*

*Beweis.* Nach Satz 3.2.1 ist das Parikh-Bild einer kontextfreien Sprache eine semi-lineare Menge, diese Eigenschaft kann leicht auf das erweiterte Parikhbild übertragen werden. Damit ist dann das erweiterte Parikh-Bild der Sprache der Automatenkomponente eines Parikh-Automaten,  $\Phi(\mathcal{L}(\mathcal{A}))$  für  $(\mathcal{A}, C)$  ebenfalls semi-linear, da dies sogar eine reguläre Sprache ist. Es gilt:

$$\mathcal{L}((\mathcal{A}, C)) \neq \emptyset \Leftrightarrow \Phi(\mathcal{L}(\mathcal{A})) \cap C \neq \emptyset.$$

Das erweiterte Parikh-Bild der Sprache der Automatenkomponente muss also Vektoren der Menge  $C$  enthalten. Da der Schnitt von semi-linearen Mengen effektiv berechenbar ist, ist damit auch das Nichtleerheitsproblem entscheidbar.  $\square$

Wie oben erwähnt, ist das Parikh-Bild einer regulären Sprache semi-linear. Dies führt zu der praktischen Eigenschaft, dass das Parikh-Bild einer solchen Sprache nach Satz 3.1.3 durch eine Presburger-Formel mit einer freien Variablen für jedes Symbol des Alphabets beschrieben werden kann, was in der Folge für den Beweis der Entscheidbarkeit verschiedener Probleme genutzt wird. Bezüglich der Größe einer solchen Presburger-Formel zeigen Seidl, Schwentick und Muscholl den folgenden Satz [34, 35].

**Satz 3.2.3** (Seidl, Schwentick, Muscholl). *Zu jeder regulären Sprache  $L$  kann eine Presburger-Formel  $\varphi_L$  in existentieller Normalform für das Parikh-Bild von  $L$  in linearer Zeit berechnet werden.*

### 3.3 Baumautomaten mit lokalen Anzahlbedingungen

Wir werden hier verschiedene Automatenmodelle vorstellen, die ausschließlich Anzahlbedingungen an die Kinder eines Knotens in einem Baum stellen. Wie eingangs erwähnt, werden solche Bedingungen in dieser Arbeit *lokale Anzahlbedingungen* genannt. Sie werden für unbeschränkt verzweigte Bäume definiert, also über dem Beschriftungsalphabet einer Baumsprache bzw. über den Zuständen eines passenden Baumautomaten. Es ist eine natürliche Vorgehensweise, genau solche Bedingungen zu stellen, da im Kontext der regulären, unbeschränkt verzweigten Baumsprachen die Beschriftungen der Kinder eines Knotens eben einer Regularität genügen müssen. Dies heißt praktisch, dass sie von links nach recht gelesen ein Wort bilden, das in der Sprache eines regulären Ausdrucks über dem gegebenen Beschriftungsalphabet liegt.

Um die Anzahlen von Symbolen in einem solchen Wort in einer bestimmten Form beschränken zu können, betrachten wir hier zwei grundlegende Ansätze für das Ausführen von Transitionen lokal zählender Baumautomaten:

**Presburger-Baumautomat:** Reguläre Ausdrücke werden mit Presburger-Formeln kombiniert; die Kardinalitäten von Symbolen eines Wortes müssen diese Formeln dann erfüllen und das Wort muss in der Sprache des Ausdruckes sein.

**Parikh-Baumautomat:** Man verfügt über Parikh-Wortautomaten, die die Kinder eines Knotens als ein Wort einlesen und es je nach kodierten Anzahlbedingungen akzeptieren oder verwerfen.

Beide Ansätze können sowohl über dem Beschriftungsalphabet als auch über den Zuständen eines Baumautomaten definiert werden; wir werden im Folgenden alle Möglichkeiten genau betrachten und vergleichen.

### 3.3.1 Presburger-Baumautomaten

Die in diesem Kapitel vorgestellten Automaten wurden im Wesentlichen von Seidl, Schwentick und Muscholl entwickelt [33, 34, 35]. Beonoit Groz erweiterte sie gemeinsam mit Wolfgang Thomas und Wong Karianto, dies ist in [16] nachzulesen.

Die erweiterten Baumautomatenmodelle nutzen - wie für unbeschränkt verzweigte Bäume üblich - reguläre Sprachen in ihren Transitionen, die von den Kindern eines Knotens erfüllt werden müssen. Der grundlegende Aufbau dieses Modells folgt dabei der in Kapitel 2 vorgestellten Definition 2.3.4 eines unbeschränkt verzweigten Baumautomaten. Die Transitionsrelation eines solchen Automaten  $\mathfrak{A} = (Q, \Sigma, \Delta, F)$  ist von der Form

$$\Delta \subseteq \text{Reg}(Q) \times \Sigma \times Q .$$

Die regulären Ausdrücke aus  $\text{Reg}(Q)$  sollen jetzt um Anzahlbedingungen erweitert werden. Dies geschieht durch die Boolesche Kombination regulärer Ausdrücke und Presburger-Formeln. Die Formeln können entweder über der Zustandsmenge des Automaten oder über den Symbolen des Alphabets definiert werden. Wir betrachten zunächst Formeln über den Zuständen. Für jeden Zustand  $q$  des Automaten soll es dabei eine freie Variable  $y_q$  geben, deren Semantik "Anzahl an Symbolen  $q$ " ist. Sei daher die Menge  $Y_Q$  durch

$$Y_Q = \{y_q \mid q \in Q\}$$

definiert; jedem Zustand ist eindeutig eine Variable zugeordnet Diese Menge nutzen wir zur Definition von Kombinationen regulärer Ausdrücke und Presburger-Formeln.

**Definition 3.3.1.** Ein Presburger-regulärer Ausdruck über  $Q$  ist die Boolesche Kombination regulärer Ausdrücke über  $Q$  und quantorenfreier Presburger Formeln mit freien Variablen aus der Menge  $Y_Q$ . Mit  $\text{PReg}(Q)$  ist die Menge aller Presburger-regulären Ausdrücke über dem Alphabet  $Q$  bezeichnet.

Intuitiv ist ein Wort  $w$  in der Sprache eines Presburger-regulären Ausdrucks, wenn das Wort die gegebene Boolesche Kombination von regulären Ausdrücken und Presburger-Formeln erfüllt. Wir schreiben dafür  $w \models \psi$ .

**Beispiel 3.3.1.** Seien  $Q = \{p, q\}$ ,  $Y_Q = \{y_p, y_q\}$ ,  $\psi = p^*q \wedge y_p > y_q \in PReg(Q)$  ein Presburger-regulärer Ausdruck und  $w = pppq$  ein Wort über  $Q$ . Vorausgesetzt, dass auf dem Alphabet eine Ordnung herrscht, ergibt sich für  $w$  das Parikh-Bild  $\Phi(w) = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$ . Es ist offensichtlich, dass  $w$  in der Sprache des Presburger-regulären Ausdrucks liegt, da es zum einen den regulären Ausdruck  $p^*q$  erfüllt, und zum anderen sein Parikh-Bild eine Interpretation von  $y_p > y_q$  ist.

Presburger-reguläre Ausdrücke sollen als Bedingungen für das Ausführen einer Transition eines Baumautomaten dienen. So stellt sich natürlich die Frage nach der Entscheidbarkeit dieser Ausdrücke. Seidl, Schwentick und Muscholl zeigen in [33], dass das Wortproblem entscheidbar ist. In [34] zeigen sie dann die PSPACE-Vollständigkeit. Den Beweis für die Entscheidbarkeit geben wir hier an, da er die Grundlage einiger folgender Beweise bilden wird.

**Satz 3.3.1.** *Es ist für einen Presburger-regulären Ausdruck  $\psi$  über  $Q$  oder  $\Sigma$  entscheidbar, ob es ein Wort  $w$  gibt, sodass  $w \models \psi$  gilt.*

*Beweis.* Zunächst wird die Form, in der ein Ausdruck vorliegen soll, festgelegt; dies soll keine Einschränkung darstellen.

**Lemma 3.3.1.** *Jeder Presburger-reguläre Ausdruck kann in einen äquivalenten Ausdruck in disjunktiver Normalform umgewandelt werden.*

Die Korrektheit dieses Lemmas ist offensichtlich, da jede Presburger-Formel in eine äquivalente, quantorenfreie Formel in disjunktiver Normalform transformiert werden kann, wie Abschnitt 3.1 zu entnehmen ist. Die resultierenden Teilformeln sind dann entsprechend mit regulären Ausdrücken verknüpft.

Ein Presburger-regulärer Ausdruck ist also von der Form

$$\psi = \bigvee_i \bigwedge_j \psi_{ij}.$$

Jeder Teilausdruck  $\psi_{ij}$  ist entweder ein regulärer Ausdruck oder eine Presburger-Formel. Man kann die Teilausdrücke so strukturieren, dass man einen Ausdruck  $\bigvee_i (r_i \wedge \pi_i)$  erhält, wobei die  $r_i$  reguläre Ausdrücke und die  $\pi_i$  Presburger-Formeln sind. Der gesamte Ausdruck ist also erfüllbar genau dann, wenn eine der Konjunktionen  $r_i \wedge \pi_i$  erfüllbar ist. Um die Erfüllbarkeit einer solchen Konjunktion zu überprüfen, wird das Parikh-Bild

benutzt. Nach Satz 3.2.3 kann zu einem gegebenen regulären Ausdruck  $r$  eine Presburger-Formel  $\pi_r$  berechnet werden, die das Parikh-Bild von  $r$  beschreibt, der Menge aller Parikh-Bilder von Wörtern aus der Sprache von  $r$ . Diese Formel hat dann freie Variablen aus der Menge  $Y_Q$  bzw.  $Y_\Sigma$ .

Es muss ein Wort  $w$  aus der Sprache von  $r_i$  geben, dessen Parikh-Bild eine Interpretation von  $\pi_i$  ist,  $r_i \wedge \pi_i$  ist also erfüllbar genau dann, wenn die Presburger-Formel  $\pi_r \wedge \pi_i$  erfüllbar ist. Nach Satz 3.1.2 ist dies entscheidbar.  $\square$

Mit diesen Voraussetzungen kann das erste Modell eines zählenden Baumautomaten definiert werden. Wie schon vorher erwähnt, betrachten wir zunächst einen Automaten, der Anzahlbedingungen an seine erreichten Zustände stellen kann. In unserem Kontext heißt das, dass lokale Presburger-Bedingungen an den Lauf eines Automaten auf einem bestimmten Eingabebaum gestellt werden.

**Definition 3.3.2.** Ein lokal über Zuständen zählender Presburger-Baumautomat ist ein Tupel  $\mathfrak{A}_{Pres} = (Q, \Sigma, \Delta, F, \Phi)$  mit einer endlichen Zustandsmenge  $Q$ , einem Beschriftungsalphabet  $\Sigma$ , einer Menge von akzeptierenden Zuständen  $F$  und einer endlichen Menge von Presburger-Formeln  $\Phi$  mit Presburger-Formeln über  $Y_Q$ . Die Transitionsrelation  $\Delta$  ist definiert durch

$$\Delta \subseteq PReg(Q) \times \Sigma \times Q$$

mit Presburger-Formeln nur aus  $\Phi$ .

Ein Lauf eines solchen Automaten ist gegeben durch eine Funktion  $\rho : dom_t \rightarrow Q$ , sodass es für alle  $x \in dom_t$  und deren Nachfolger  $x_1, \dots, x_n$  einen Presburger-regulären Ausdruck  $\psi$  gibt, sodass

$$\rho(x_1) \dots \rho(x_n) \models \psi \text{ und } (\psi, t(x), \rho(x)) \in \Delta$$

gilt. Ein Lauf  $\rho$  auf einem Baum  $t$  ist akzeptierend, wenn seine Wurzel mit einem akzeptierenden Zustand beschriftet ist. Wir nennen diesen Automaten in der Folge vereinfachend *Presburger-Baumautomat*, die anderen Automatenmodelle mit Presburger-Bedingungen werden entsprechend bezeichnet.

Ein ähnliches Automatenmodell benutzt statt Presburger-regulärer Ausdrücke über Zuständen solche über dem Alphabet des Automaten. Die Presburger-Formeln in den Ausdrücken haben freie Variablen über der Menge  $Y_\Sigma$  mit

$$Y_\Sigma = \{y_a \mid a \in \Sigma\}.$$

Analog dazu definieren wir diesen Automaten mit Anzahlbedingungen über den Symbolen des Alphabets.

**Definition 3.3.3.** Ein lokal über Symbolen zählender Presburger-Baumautomat ist ein Tupel  $\mathfrak{A}_{Pres} = (Q, \Sigma, \Delta, F, \Phi)$  mit einer endlichen Zustandsmenge  $Q$ , einem Beschriftungsalphabet  $\Sigma$ , einer Menge von akzeptierenden Zuständen  $F$  und einer endlichen Menge von Presburger-Formeln  $\Phi$  mit Presburger-Formeln über  $Y_\Sigma$ . Die Transitionrelation  $\Delta$  ist definiert durch

$$\Delta \subseteq PReg(Q) \times \Sigma \times Q$$

mit Presburger-Formeln nur aus  $\Phi$ .

Die Presburger-regulären Ausdrücke der Transitionen seien o.B.d.A. alle von der Form  $\psi = r \wedge \pi$ . Das ist keine Einschränkung, da nach Lemma 3.3.1 jeder Ausdruck in disjunktive Normalform umgewandelt werden kann, von den Disjunktionen muss dann genau eine Konjunktion  $r \wedge \pi$  erfüllt werden. Ein Lauf des Automaten ist definiert durch  $\rho : dom_t \rightarrow Q$ , sodass es für alle  $x \in dom_t$  und deren Nachfolger  $x_1 \dots x_n \in dom_t$  einen Presburger-regulären Ausdruck  $\psi = r \wedge \pi$  gibt, sodass

$$\rho(x_1) \dots \rho(x_n) \in \mathcal{L}(r) \text{ und } t(x_1) \dots t(x_n) \models \pi \text{ und } (\psi, t(x), \rho(x)) \in \Delta$$

gilt. Ein Lauf  $\rho$  auf einem Baum  $t$  ist akzeptierend, wenn seine Wurzel mit einem akzeptierenden Zustand beschriftet ist. Seidl, Schwentick und Muscholl zeigten die folgenden Resultate für die wichtigen Entscheidungsprobleme.

**Satz 3.3.2** (Seidl, Schwentick, Muscholl). *Für Presburger-Baumautomaten mit lokalen Anzahlbedingungen sowohl über Zuständen als auch über Symbolen sind Nichtleerheit und Zugehörigkeit entscheidbar. Universalität ist unentscheidbar.*

Die Entscheidbarkeit des Nichtleerheits- und des Zugehörigkeitsproblems ergibt sich als logische Konsequenz der Entscheidbarkeit dieser Probleme für Baumautomaten ohne Anzahlbedingungen und der für Presburger-reguläre Ausdrücke. Die Unentscheidbarkeit des Universalitätsproblems für Baumautomaten mit Presburger-Formeln wird in [33] durch eine Reduktion des Halteproblems für 2-Zähler-Automaten mit leerer Eingabe bewiesen.

### 3.3.2 Parikh-Baumautomaten

In diesem Kapitel werden lokal zählende Baumautomaten vorgestellt, die eine Erweiterung der Parikh-Automaten aus Kapitel 3.2 sind. Diese Automaten wurden von Groz in [16] vorgestellt. Auch hier sollen als Bedingung für das Ausführen einer Bottom-up-Transition die Kinder eines Knotens ein Wort aus einer bestimmten regulären Sprache bilden und zusätzlich einer Anzahlbedingung genügen. Dies wird hier durch Parikh-Automaten in den Transitionen realisiert. Die Bedingung für eine Transitionsausführung

ist also ein akzeptierender Lauf eines Parikh-Automaten. Anzahlbedingungen werden sowohl über den Zuständen des Automaten als auch über den Symbolen des Eingabealphabetes definiert.

**Definition 3.3.4.** Ein lokal über Zuständen zählender Parikh-Baumautomat ist ein Tupel  $\mathfrak{A} = (Q, \Sigma, \Delta, F, \Phi)$  mit einer endlichen Zustandsmenge  $Q$ , einem Beschriftungsalphabet  $\Sigma$ , einer Transitionsrelation  $\Delta$ , einer Menge von akzeptierenden Zuständen  $F$ .  $\Phi$  ist eine Menge von Parikh-Automaten auf Wörtern über  $Q \times D$  für ein  $D \in \mathbb{N}^n, n \in \mathbb{N}$ .  $\Delta$  ist definiert durch

$$\Delta \subseteq \Phi \times \Sigma \times Q.$$

Ein Lauf eines des Automaten auf einem Baum  $t$  ist gegeben durch eine Funktion  $\rho : \text{dom}_t \rightarrow Q$ , sodass es für alle Knoten  $x \in \text{dom}_t$  und deren Nachfolger  $x_1 \dots x_n \in \text{dom}_t$  einen Parikh-Automaten  $(\mathfrak{A}, C) \in \Phi$  und Vektoren  $d_1, \dots, d_n \in D$  gibt, sodass

$$(\rho(x_1), d_1) \dots (\rho(x_n), d_n) \in \mathcal{L}(\mathfrak{A}, C) \text{ und } ((\mathfrak{A}, C), t(x), \rho(x)) \in \Delta$$

gilt. Ein Lauf  $\rho$  auf einem Baum  $t$  ist akzeptierend, wenn seine Wurzel mit einem akzeptierenden Zustand beschriftet ist. Anzahlbedingungen werden durch die semi-linearen Constraint-Mengen der Parikh-Automaten definiert. Parikh-Baumautomaten, die auf den Symbolen des Eingabealphabetes zählen, werden analog definiert.

**Definition 3.3.5.** Ein lokal über Symbolen zählender Parikh-Baumautomat ist ein Tupel  $\mathfrak{A} = (Q, \Sigma, \Delta, F, \Phi)$  mit einer endlichen Zustandsmenge  $Q$ , einem Beschriftungsalphabet  $\Sigma$ , einer Transitionsrelation  $\Delta$  und einer Menge von akzeptierenden Zuständen  $F$ .  $\Phi$  ist eine Menge von Parikh-Automaten auf Wörtern über  $\Sigma \times D$  für ein  $D \in \mathbb{N}^n, n \in \mathbb{N}$ .  $\Delta$  ist definiert durch

$$\Delta \subseteq \Phi \times \Sigma \times Q.$$

Ein Lauf auf einem Baum  $t$  ist gegeben durch eine Funktion  $\rho : \text{dom}_t \rightarrow Q$ , sodass es für alle Knoten  $x \in \text{dom}_t$  und deren Nachfolger  $x_1 \dots x_n \in \text{dom}_t$  einen Parikh-Automaten  $(\mathfrak{A}, C) \in \Phi$  und Vektoren  $d_1, \dots, d_n \in D$  gibt, sodass

$$(t(x_1), d_1) \dots (t(x_n), d_n) \in \mathcal{L}(\mathfrak{A}, C) \text{ und } ((\mathfrak{A}, C), t(x), \rho(x)) \in \Delta$$

gilt. Ein Lauf  $\rho$  auf einem Baum  $t$  ist akzeptierend, wenn seine Wurzel mit einem akzeptierenden Zustand beschriftet ist.

**Satz 3.3.3.** *Das Nichtleerheitsproblem für Parikh-Baumautomaten mit lokalen Anzahlbedingungen ist entscheidbar.*



Auch hier werden die Konzepte *Presburger-Zählen* und *Parikh-Zählen* zur Definition von entsprechenden Automatenmodellen genutzt.

### 3.4.1 Presburger-Baumautomaten mit globalen Anzahlbedingungen

Der Ansatz für diese Automaten ist, einen Automaten auf unbeschränkt verzweigten Bäumen um eine Presburger-Formel zu erweitern, der die gesamte Beschriftung eines Eingabebaumes genügen muss. Dies ist technisch möglich, da eine Instanz einer unbeschränkten verzweigten Baumsprache, ein Baum, in endlicher Form vorliegt. Es könnte beispielsweise die gesamte Beschriftung des Baumes in einer In-Order-Traversierung ausgelesen und als Wort gespeichert werden. Da hier nur die Anzahlen der Symbole, nicht aber ihre Struktur von Interesse sind, genügt die Überprüfung, ob dieses Wort die Presburger-Formel erfüllt. Dies soll Teil der Akzeptierbedingung der Automaten sein.

Um eine Erfüllbarkeitsrelation zwischen Bäumen und Presburger-Formeln zu definieren, sei in der Folge das endliche Wort  $w_t$  für einen Baum  $t$  das Ergebnis einer In-Order-Traversierung von  $t$ , in dem bei Besuchen eines jeden Knotens dessen Beschriftung mit den bisherigen Beschriftungen konkateniert wurde.

**Definition 3.4.1.** Seien ein unbeschränkt verzweigter Baum  $t$  über dem Beschriftungsalphabet  $\Sigma$  und eine Presburger-Formel  $\psi$  mit freien Variablen aus der Menge  $Y_\Sigma = \{y_a \mid a \in \Sigma\}$  gegeben. Der Baum  $t$  erfüllt die Presburger-Formel  $\psi$ , kurz  $t \models \psi$ , genau dann, wenn das Parikh-Bild von  $w_t$  eine Interpretation von  $\psi$  ist.

Groz definierte global zählende Presburger-Automaten in [16] als Erweiterung der Presburger-Baumautomaten nach Definition 3.3.2. Man erhält Automaten, die sowohl lokal als auch global zählen können. Wir übernehmen diese Definition und werden zusätzlich einen Algorithmus zur Lösung des Nichtleerheitsproblems angeben.

**Definition 3.4.2.** Ein lokal und global auf Zuständen zählender Presburger-Baumautomat ist ein Tupel  $\mathfrak{A}_{Pres} = (Q, \Sigma, \Delta, F, \Phi, \vartheta)$  mit einer endlichen Zustandsmenge  $Q$ , einem Beschriftungsalphabet  $\Sigma$ , einer Menge von akzeptierenden Zuständen  $F$ , einer endlichen Menge von Presburger-Formeln  $\Phi$  mit Presburger-Formeln über  $Y_Q$  und einer Presburger-Formel  $\vartheta$  über  $Y_Q$ . Die Transitionsrelation  $\Delta$  ist definiert durch

$$\Delta \subseteq PReg(Q) \times \Sigma \times Q$$

mit Presburger-Formeln nur aus  $\Phi$ .

Ein Lauf eines solchen Automaten ist gegeben durch eine Funktion  $\rho : dom_t \rightarrow Q$ , sodass es für alle  $x \in dom_t$  und deren Nachfolger  $x_1, \dots, x_n$  einen Presburger-regulären

Ausdruck  $\psi$  gibt, sodass

$$\rho(x_1) \dots \rho(x_n) \models \psi \text{ und } (\psi, t(x), \rho(x)) \in \Delta$$

gilt. Ein Lauf  $\rho$  auf einem Baum  $t$  ist *akzeptierend*, wenn seine Wurzel mit einem akzeptierenden Zustand beschriftet ist und  $r \models \vartheta$  gilt, er also die globale Presburger-Formel erfüllt.

Analog kann auch hier natürlich wieder ein Automat definiert werden, der lokal und global auf den Symbolen des Beschriftungsalphabet zählt. Wir geben auch hierfür die vollständige Definition an.

**Definition 3.4.3.** Ein lokal und global auf Symbolen zählender Presburger-Baumautomat ist ein Tupel  $\mathfrak{A}_{Pres} = (Q, \Sigma, \Delta, F, \Phi, \vartheta)$  mit einer endlichen Zustandsmenge  $Q$ , einem Beschriftungsalphabet  $\Sigma$ , einer Menge von akzeptierenden Zuständen  $F$ , einer endlichen Menge von Presburger-Formeln  $\Phi$  mit Presburger-Formeln über  $Y_\Sigma$  und einer Presburger-Formel  $\vartheta$  über  $Y_\Sigma$ . Die Transitionsrelation  $\Delta$  ist definiert durch

$$\Delta \subseteq PReg(Q) \times \Sigma \times Q$$

mit Presburger-Formeln nur aus  $\Phi$ .

Ein Lauf eines solchen Automaten ist gegeben durch eine Funktion  $\rho : dom_t \rightarrow Q$ , sodass es für alle  $x \in dom_t$  und deren Nachfolger  $x_1, \dots, x_n$  einen Presburger-regulären Ausdruck  $\psi$  gibt, sodass

$$t(x_1) \dots t(x_n) \models \psi \text{ und } (\psi, t(x), \rho(x)) \in \Delta$$

gilt. Ein Lauf  $\rho$  auf einem Baum  $t$  ist *akzeptierend*, wenn seine Wurzel mit einem akzeptierenden Zustand beschriftet ist und  $t \models \vartheta$  gilt, der Eingabebaum also die globale Presburger-Formel erfüllt.

Groz zeigte die Entscheidbarkeit des Nichtleerheitsproblems [16].

**Satz 3.4.1** (Groz). *Das Nichtleerheitsproblem für lokal und global zählende Presburger-Baumautomaten ist entscheidbar.*

### 3.4.2 Parikh-Baumautomaten mit globalen Anzahlbedingungen

Auch das Parikh-Zählen wird zur Definition globaler Anzahlbedingungen genutzt. Dazu stellten Klaedtke und Rueß einen rein global zählenden Parikh-Baumautomaten vor [20], wir werden hier aber auch analog zu den Presburger-Automaten ein Modell definieren, das sowohl global als auch lokal zählen kann. Zunächst betrachten wir das Modell von Klaedtke und Rueß.

**Definition 3.4.4.** Ein global zählender Parikh-Baumautomat der Dimension  $N \geq 1$  ist ein Paar  $(\mathfrak{A}, \mathcal{C})$ , wobei  $\mathfrak{A}$  ein unbeschränkt verzweigter Baumautomat  $\mathfrak{A} = (Q, \Sigma \times D, \Delta, F)$  über dem Alphabet  $\Sigma \times D$  ist und  $\mathcal{C}$  eine semi-lineare Teilmenge von  $\mathbb{N}^n$ .  $D$  ist eine nicht leere, endliche Teilmenge von  $\mathbb{N}^n$ .

Dieser Automat besteht wie der Parikh-Automat auf Wörtern nach Definition 3.2.4 aus einer Automatenkomponente und einer *Constraint*-Menge. Die Funktionsweise entspricht der des Wortautomaten, in den Transitionen wird für jede Beschriftung ein entsprechender Vektor aus der Menge  $D$  addiert. So erhält man am Ende eines Laufes das erweiterte Parikh-Bild eines Baumes, dieser Vektor muss in der Constraint-Menge  $\mathcal{C}$  enthalten sein.

**Definition 3.4.5.** Ein lokal und global zählender Parikh-Baumautomat ist ein Paar  $(\mathfrak{A}, \mathcal{C})$ , wobei  $\mathfrak{A}$  ein lokal zählender Parikh-Baumautomat  $\mathfrak{A} = (Q, \Sigma, \Delta, F, \Phi)$  nach Definition 3.3.4 ist und  $\mathcal{C}$  eine semi-lineare Teilmenge von  $\mathbb{N}^n$ .  $D$  ist eine nicht leere, endliche Teilmenge von  $\mathbb{N}^n$ .

Dieser Automat benutzt Parikh-Wortautomaten in seinen Transitionen, die lokal die Kinder von Knoten zählen und eine globale Constraint-Menge  $\mathcal{C}$ , die durch schrittweise Addition von Vektoren aus  $D$  verifiziert wird.

Wir verzichten hier auf die unterschiedlichen Definitionen von Automaten, die auf Zuständen bzw. auf Symbolen zählen, da dies analog zu den lokal zählenden Baumautomaten ist.

### 3.5 Vergleich der Modelle

In diesem Kapitel wird ein kurzer Überblick über die Aussagestärke der hier vorgestellten Modelle gegeben. Einige der Beweise sind in [16] nachzulesen, andere wurden hier formuliert. Mit den Resultaten teilen wir die durch die verschiedenen Automaten definierten, zählenden Baumsprachen in fünf Klassen ein und ordnen diese hierarchisch.

Im Gegensatz zu Automaten über Wörtern definieren lokal zählende Presburger-Baumautomaten und Parikh-Baumautomaten dieselben Sprachen.

**Satz 3.5.1.** *Presburger-Baumautomaten nach Definition 3.3.2 und Parikh-Baumautomaten nach Definition 3.3.4 haben die gleiche Aussagekraft.*

*Beweis.*

**Lemma 3.5.1.** *Für jeden Presburger-Baumautomaten  $\mathfrak{A}$  gibt es einen Parikh-Baumautomaten  $\mathfrak{B}$  sodass  $t \in \mathcal{L}(\mathfrak{A}) \Rightarrow t \in \mathcal{L}(\mathfrak{B})$ .*

*Beweis.* Zu dem gegebenen Presburger-Baumautomaten  $\mathfrak{A} = (Q_1, \Sigma, \Delta_1, F_1, \Phi_1)$  wird ein Parikh-Baumautomat  $\mathfrak{B} = (Q_2, \Sigma, \Delta_2, F_2, \Phi_2)$  konstruiert. Die Transitionsrelation des Presburger-Baumautomaten ist gegeben durch

$$\Delta_1 \subseteq PReg \times \Sigma \times Q .$$

Für jede Transition  $(r, a, q) \in \Delta_1$  führen wir die folgende Änderung durch: Jeder Presburger-reguläre Ausdruck  $r \in PReg(Q)$  kann nach Lemma 3.3.1 in disjunktive Normalform umgewandelt werden. Es gilt also

$$r = r_1 \wedge \varphi_1 \vee \dots \vee r_n \wedge \varphi_n .$$

Wir bilden für jeden regulären Ausdruck  $r_i, 1 \leq i \leq n$ , einen nichtdeterministischen endlichen Automaten  $\mathcal{A}_i$  und bilden aus diesen Automaten den Vereinigungsautomaten  $\mathcal{A} = \bigcup_i \mathcal{A}_i$ . Für jede Formel  $\varphi_i$  berechnen wir nach Satz 3.1.3 eine semi-lineare Menge  $C_i$  und bilden die Menge  $C = \bigcup_i C_i$ . Wir ersetzen in der Transition  $(r, a, q)$  den regulären Ausdruck  $r$  durch den Parikh-Wortautomaten  $(\mathcal{A}, C)$  und fügen die resultierende Transition der Menge  $\Delta_2$  und  $(\mathcal{A}, C)$  der Menge  $\Phi_2$  hinzu. Weiterhin gilt:

- $Q_2 = Q_1$
- $F_2 = F_1$

Die Parikh-Wortautomaten in den Transitionen erkennen jeweils die gleichen Wortsprachen wie die Presburger-regulären Ausdrücke, daher akzeptiert  $\mathfrak{B}$  alle Bäume, die  $\mathfrak{A}$  akzeptiert.  $\square$

**Lemma 3.5.2** (Groz). *Für jeden Parikh-Baumautomaten  $\mathfrak{B}$  gibt es einen Presburger-Baumautomaten  $\mathfrak{A}$  sodass  $t \in \mathcal{L}(\mathfrak{B}) \Rightarrow t \in \mathcal{L}(\mathfrak{A})$ .*

Der Beweis zu diesem Lemmas ist in [16] nachzulesen. Die Idee ist die Konstruktion eines Presburger-Baumautomaten, in dessen Zuständen alle möglichen Vektoren der Parikh-Wortautomaten der Transitionen kodiert werden.

Mit Lemma 3.5.1 und Lemma 3.5.2 folgt Satz 3.5.1.  $\square$

Für global auf Symbolen zählende Baumautomaten gilt diese Eigenschaft nicht:

**Satz 3.5.2.** *Lokal und global auf Symbolen zählende Presburger-Baumautomaten sind von geringerer Ausdrucksstärke als lokal und global auf Symbolen zählende Parikh-Baumautomaten.*

*Beweis.*

**Lemma 3.5.3.** *Für jeden lokal und global zählenden Presburger-Baumautomaten  $\mathfrak{A}_{Pres}$  gibt es einen lokal und global zählenden Parikh-Baumautomaten  $(\mathfrak{A}_{Par}, C)$ , sodass:*

$$\forall t \in T_\Sigma : t \in T(\mathfrak{A}_{Pres}) \Leftrightarrow t \in T(\mathfrak{A}_{Par})$$

*Beweis.* Sei  $\mathfrak{A}_{Pres} = (Q, \Sigma, \Delta, F, \Phi, \vartheta)$  gegeben. Wir konstruieren aus  $\mathfrak{A}_{Pres}$  einen lokal und global zählenden Parikh-Baumautomaten  $(\mathfrak{A}_{Par}, C)$  mit  $\mathfrak{A}_{Par} = (Q_1, \Sigma_1, \Delta_1, F_1, \Phi_1)$ . Dabei wird wie im Beweis zu Lemma 3.5.1 unter Vernachlässigung der globalen Presburger-Formel  $\vartheta$  aus  $\mathfrak{A}_{Pres}$  ein lokal zählender Parikh-Baumautomat  $\mathfrak{A}_{Par}$  konstruiert. Dann berechnen wir nach Satz 3.1.3 aus  $\vartheta$  eine semi-lineare Menge  $C$  und erhalten damit den Automaten  $(\mathfrak{A}_{Par}, C)$ .  $\square$

**Lemma 3.5.4.** *Es gibt Baumsprachen, die von lokal und global auf Symbolen zählenden Parikh-Baumautomaten erkannt werden, aber nicht von lokal und global auf Symbolen zählenden Presburger-Baumautomaten.*

*Beweis.* Zu einem Wort  $w = a_1 \dots a_n \in \Sigma^*$  sei der unäre Baum  $t_w$  gegeben durch  $t_w = a_1(a_2(\dots(a_n)\dots))$ . Sei  $\mathfrak{A}$  ein global zählender Presburger-Baumautomat, der nur Sprachen von unären Bäumen akzeptiert. Es ist offensichtlich, dass  $\mathfrak{A}$  einem Wortautomaten  $\mathcal{A}$  mit Presburger-Bedingung entspricht, der genau die zu den unären Bäumen gehörigen Wörter akzeptiert. Analog dazu entspricht ein Parikh-Baumautomat auf unären Bäumen einem Parikh-Wortautomaten. Wir können also zu jedem zählenden Wortautomaten  $\mathcal{A}$  einen zählenden Baumautomaten  $\mathfrak{A}$  auf unären Bäumen konstruieren, sodass

$$\forall w \in \Sigma^* : w \in \mathcal{L}(\mathcal{A}) \Leftrightarrow t_w \in \mathcal{T}(\mathfrak{A})$$

Es sei angenommen, dass Parikh-Wortautomaten und Wortautomaten mit Presburger-Bedingung von gleicher Aussagekraft sind. Sei die Sprache  $L = \{a^n b^n a^m c^m \mid n, m \in \mathbb{N}^+\}$  durch den Parikh-Automaten aus Beispiel 3.2.2 gegeben. Diese Sprache ist nicht von einem Wortautomaten mit einem Presburger-regulären Ausdruck über den Symbolen des Alphabets erkennbar. Wir verzichten auf einen formalen Beweis, da die Korrektheit dieser Aussage offensichtlich ist: Ein Presburger-regulärer Ausdruck kann lediglich eine Anzahlbedingung über die Anzahl von Symbolen im gesamten Wort definieren. Das Symbol  $a$  muss hier aber zunächst in gleicher Anzahl wie das Symbol  $b$  auftreten, anschließend in gleicher Anzahl wie das Symbol  $c$ . Dies ist ein Widerspruch zu der Annahme.

Wir haben gezeigt, dass Parikh-Wortautomaten von größerer Aussagekraft sind als Wortautomaten mit Presburger-Bedingung. Dann sind auch lokal und global auf Symbolen zählende Parikh-Baumautomaten von größerer Aussagekraft als lokal und global

auf Symbolen zählende Presburger-Baumautomaten jeweils auf unären Bäumen. Damit gilt diese Eigenschaft auch im Allgemeinen für die beiden Automaten.  $\square$

Mit den beiden Lemmas ist Satz 3.5.2 gezeigt.  $\square$

Der im vorangegangenen Satz gezeigte Unterschied gilt nicht für lokal und global auf Zuständen der Automaten zählende Automaten, da wie im Beweis zu Satz 3.5.1 dann gezeigt werden kann, dass die beiden Modelle gleich mächtig sind:

**Korollar 3.5.1.** *Lokal und global auf Zuständen zählende Presburger- und Parikh-Baumautomaten haben die gleiche Aussagekraft.*

Um den Unterschied zwischen auf Symbolen und auf Zuständen zählenden Baumautomaten herauszustellen, betrachten wir die Baumsprache  $\mathcal{T}$ , die alle Bäume über dem Alphabet  $\{a\}$  beinhaltet, deren Knoten insgesamt öfter 3 Nachfolger als 2 Nachfolger haben. Ein auf Symbolen zählender Baumautomat kann diese Sprache nicht erkennen, da nur die Anzahl von Symbolen gezählt werden kann: Knoten, die mit  $a$  beschriftet sind, können mit einer globalen Bedingung nicht voneinander unterschieden werden. Zählt man auf  $n$ , ist diese Sprache definierbar:

Sei  $\mathfrak{A}_{\mathcal{T}} = (Q, \Sigma, \Delta, F, \Phi, \vartheta)$  ein lokal und global zählender Presburger-Automat mit Zuständen  $Q = \{q_a, q_2, q_3\}$ , in dessen Transitionsrelation Transition enthalten sind, sodass genau jeder Knoten mit 2 Nachfolgern in einem Lauf Zustand mit  $q_2$  beschriftet wird und analog ein Knoten mit 3 Nachfolgern mit  $q_3$ . Sei dazu die Presburger-Formel  $\vartheta = y_{q_3} > y_{q_2}$ . Dieser Automat definiert genau die Sprache  $\mathcal{T}$ .

Wir formulieren ohne Beweis die offensichtliche Hierarchie:

**Korollar 3.5.2.** *Auf Symbolen zählende Baumautomaten sind im Allgemeinen von geringerer Ausdruckstärke als solche, die auf Zuständen des Automaten zählen.*

Wir nutzen nun die in diesem Kapitel formulierten Ergebnisse, um die vorgestellten zählenden Baumsprachen entsprechend der Automatenmodelle in Klassen einzuteilen.

**Definition 3.5.1.** Die in Kapitel 3 vorgestellten Baumautomaten definieren die folgenden Teilklassen der Klasse **Count** zählender Baumsprachen:

- $\text{LCount}_{\text{Alph}}$ : lokal auf Symbolen des Alphabets zählende Baumsprachen
- $\text{LCount}_{\text{States}}$ : lokal auf Zuständen des Automaten zählende Baumsprachen
- $\text{PresGCount}_{\text{Alph}}$ : lokal und global auf Symbolen des Alphabets zählende Presburger-Baumsprachen

- $\text{ParikhGCount}_{\text{Alph}}$ : lokal und global auf Symbolen des Alphabets zählende Parikh-Baumsprachen
- $\text{GCount}_{\text{States}}$ : lokal und global auf Zuständen des Automaten zählende Baumsprachen

Die Vereinigung dieser Klassen definiert die Klasse  $\text{Count}$ .

Zwischen den Klassen kann keine lineare Hierarchie hergestellt werden, da  $\text{LCount}_{\text{States}}$  nicht mit  $\text{PresGCount}_{\text{Alph}}$  und  $\text{ParikhGCount}_{\text{Alph}}$  vergleichbar ist. In dem folgenden Korollar ist  $<$  semantisch durch „ist von geringerer Aussagekraft als“ zu ersetzen.

**Korollar 3.5.3.** *Für die Klassen von zählenden Baumsprachen aus Definition 3.5.1 gilt:*

- $\text{LCount}_{\text{Alph}} < \text{LCount}_{\text{States}}$
- $\text{PresGCount}_{\text{Alph}} < \text{ParikhGCount}_{\text{Alph}} < \text{GCount}_{\text{States}}$
- $\text{LCount}_{\text{Alph}} < \text{PresGCount}_{\text{Alph}}$
- $\text{LCount}_{\text{States}} < \text{GCount}_{\text{States}}$
- $\text{GCount}_{\text{States}} = \text{Count}$

# Kapitel 4

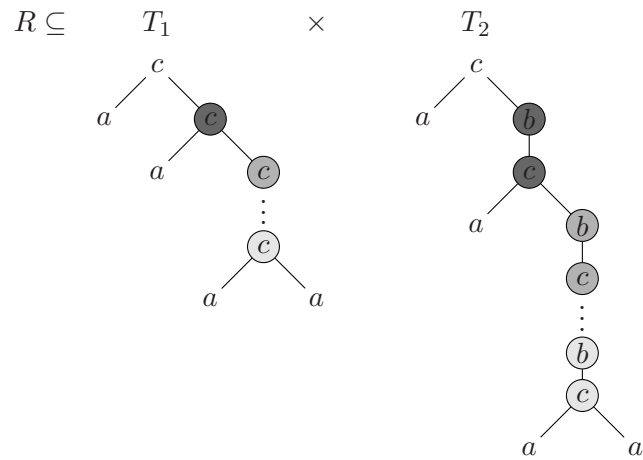
## Rationale Baumrelationen

In diesem Kapitel werden rationale Relationen über Bäumen eingeführt, die eine natürliche Erweiterung der rationalen Wortrelationen sein sollen. Es gibt verschiedene Ansätze für diese Relationen von Baumsprachen über Rangalphabeten: Raoult erweiterte 1997 in [31] die Definition rationaler Wortrelationen durch *rationale Ausdrücke mit Multivariablen* auf Baumrelationen. Bereits 1992 und auch 1997 stellte er in [30] und [31] *Multivariablengrammatiken* für rationale Baumrelationen vor. Arnold und Dauchet übertrugen 1982 durch *Bimorphismen über Bäumen* die Darstellung rationaler Wortrelationen von Nivat in [23] auf Bäume [1]. Radmacher definierte 2007 mit den *asynchronen Baumautomaten* ein Automatenmodell für diese Klasse von Relationen, das äquivalent zu den Definitionen von Raoult ist [28, 29]. Wir werden hier die rationalen Ausdrücke mit Multivariablen sowie die asynchronen Baumautomaten vorstellen.

Die Klasse der rationalen Baumrelationen wird in dieser Arbeit mit  $\mathbf{Rat}$  bezeichnet; sind explizit  $n$ -stellige Relationen gemeint, schreiben wir  $\mathbf{Rat}_n$ . Ein einführendes Beispiel soll zunächst die Beschaffenheit dieser Relationen verdeutlichen.

**Beispiel 4.0.1.** In Abbildung 4.1 ist die Relation  $R$  von zwei beschränkt verzweigten Baumsprachen  $T_1, T_2$  zu sehen. Die beiden Sprachen sind fast identisch, nur hat in dem rechten Teilbaum der zweiten Baumsprache jeder mit  $c$  beschriftete Knoten noch einen Vaterknoten, der mit  $b$  beschriftet sind.  $T_1$  und  $T_2$  stehen durch  $R$  derart in Relation, dass es für jeden mit  $c$  beschrifteten Knoten in einem Baum der Sprache  $T_1$  einen Knoten  $c$  mit Vaterknoten  $b$  in einem Baum der Sprache  $T_2$  gibt.

Ein Formalismus, der diese rationalen Relationen definiert, muss auf der einen Seite eine beliebige Synchronisation zwischen verschiedenen Bäumen ermöglichen, auf der anderen Seite muss es auch möglich sein, asynchron auf verschiedenen Bäumen zu operieren. Diese intuitive Beschreibung zeigt auf, dass genau der Ansatz der rationalen Wortrelationen übertragen werden soll. Für die hier behandelten Formalismen gilt grundsätzlich, dass sie auf unäre Bäume angewandt genau die rationalen Wortrelationen definieren. Unäre Bäume sind dabei endliche Wörter, die als Bäume kodiert sind.

Abbildung 4.1: Relation  $R$  zweier Baumsprachen  $T_1, T_2$ 

Für detailliertere Ausführungen zu Baumrelationen über beschränkt verzweigten Bäumen sei nochmals auf [28] verwiesen.

## 4.1 Multivariablen

Die mit  $\text{Rat}_n$  bezeichnete Klasse von rationalen Baumrelationen ist induktiv durch Abgeschlossenheit unter Konkatenation und Kleene Stern bezüglich Multivariablen und Vereinigung definiert. Diese ursprüngliche Definition von Raoult [31] wird hier vorgestellt.

Um eine Synchronisation zwischen verschiedenen Bäumen zu erreichen, müssen bestimmte Operationen auf Bäumen synchron ausgeführt werden. Dazu können Blätter von Bäumen mit Variablen beschriftet werden, die durch Bäume ersetzt werden. Bestimmte Variablen werden dabei so in Relation zueinander gesetzt, dass sie nur synchron ersetzt werden können. Dazu definiert Raoult die Multivariablen.

**Definition 4.1.1.** Sei  $X = \{x_1, \dots, x_n\}$  eine Menge von Variablen. Eine Multivariable  $A$  der Länge  $n > 0$  ist eine nicht leere Folge  $A = x_1 \dots x_n$  von Variablen aus  $X$ . Die Menge aller Instanzen der Multivariable  $A$  ist das kartesische Produkt  $A \times \mathbb{N}$ . Dabei ist die  $i$ -te Instanz von  $A$  gekennzeichnet durch  $A^i = x_1^i \dots x_n^i$ .

Alle Variablen einer Multivariablen müssen in einem Tupel von Bäumen grundsätzlich vollständig und genau einmal in gleicher Instanz auftreten. Einer Variablen sind durch

die Instantiierung alle anderen Variablen der Multivariablen eindeutig zugeordnet. Alle verwendeten Multivariablen sind disjunkt, haben also keine gemeinsamen Variablen.

Wir definieren die Konkatenation zweier Relationen bezüglich Multivariablen und den Kleene Stern bezüglich Multivariablen.

**Definition 4.1.2.** Seien  $R$  und  $S$  Baumrelationen.  $R$  enthalte  $n$ -Tupel und  $S$  enthalte  $m$ -Tupel von beschränkt verzweigten Bäumen. Sei  $A = x_1 \dots x_n$  eine Multivariable der Länge  $n$ . Wir definieren jeweils bezüglich einer Multivariablen  $A$ :

- die Konkatenation eines Tupels und einer Relation

$$t^{(n)} \cdot_A R := \left\{ t_1^{(n)} \mid \begin{array}{l} \text{„In } t^{(n)} \text{ werden alle Instanzen von } A \\ \text{durch jeweils genau ein Tupel aus } R \text{ ersetzt“} \end{array} \right\}$$

- die Konkatenation zweier Relationen

$$S \cdot_A R := \{ t_1^{(n)} \cdot_A S \mid t^{(n)} \in S \} .$$

- die iterierte Konkatenation

$$R^{nA} := R^{0A} \cup R \cdot_A R^{(n-1)A} \text{ mit } R^{0A} = \{(x_1^j, \dots, x_n^j)\}$$

- den Kleene Stern

$$R^{*A} := \bigcup_{n \in \mathbb{N}} R^{nA} .$$

Für die iterierte Konkatenation und den Kleene Stern muss dabei die Instanz  $j$  in der resultierenden Relation eindeutig sein.

Mit diesen Operationen können nun die rationalen Baumrelationen definiert werden:

**Definition 4.1.3.** Die Klasse  $\text{Rat}_n$  von rationalen Relationen über beschränkt verzweigten Bäumen ist die kleinste Menge, die endliche Mengen von  $n$ -Tupeln von Bäumen enthält und folgende Abschlusseigenschaften hat:

- $R \in \text{Rat}_n \wedge S \in \text{Rat}_n \Rightarrow R \cup S \in \text{Rat}_n$
- $R \in \text{Rat}_n \wedge |A| = m \wedge S \in \text{Rat}_n \Rightarrow R \cdot_A S \in \text{Rat}_n$
- $R \in \text{Rat}_n \wedge |A| = n \Rightarrow R^{*A} \in \text{Rat}_n$

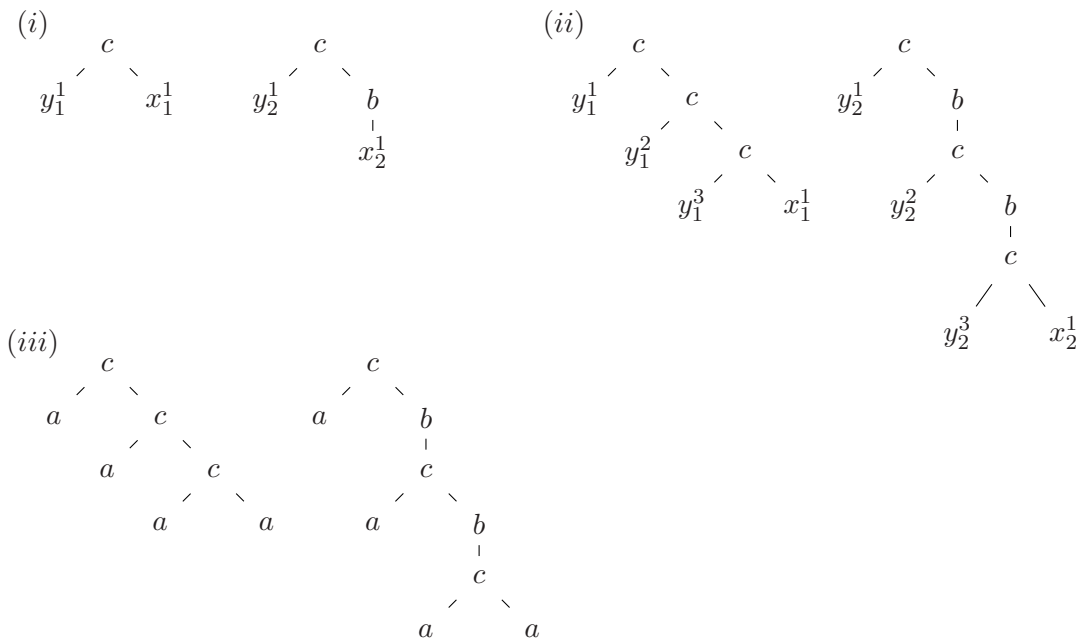


Abbildung 4.2: Erzeugung eines 2-Tupels von Bäumen durch einen rationalen Ausdruck

Durch diese Definition können rationale Baumrelationen mit rationalen Ausdrücken beschrieben werden. Um dies zu verdeutlichen, betrachten wir ein Beispiel.

**Beispiel 4.1.1.** Seien ein Alphabet  $\Sigma = \{c^2, b^1, a^0\}$  und zwei Multivariablen  $X = x_1x_2, Y = y_1y_2$  gegeben. Die rationale Baumrelation  $R$  aus Beispiel 4.0.1 ist definiert durch den rationalen Ausdruck

$$(cy_1x_1, cy_2bx_2)^{*X} \cdot_X (a, a) \cdot_Y (a, a) .$$

In Abbildung 4.2 ist in drei Schritten die Erzeugung eines Tupels  $(t_1, t_2) \in R$  durch diesen Ausdruck dargestellt. Schritt (iii) zeigt das resultierende Tupel.

Die Definition von **Rat** lässt auch die Verwendung mehrerer Variablen von ein und derselben Multivariablen in einem Baum zu, dadurch ist eine *innere Synchronisation* zwischen Teilbäumen eines Baumes möglich. Das hat zur Folge, dass nicht-reguläre Baumsprachen definierbar sind:

**Beispiel 4.1.2.** Seien  $\Sigma = \{c^2, b^1, a^0\}$ ,  $X = x_1x_2$  und  $Y = y_1y_2$  gegeben. Der rationale Ausdruck

$$(cx_1x_2) \cdot_X (by_1, by_2)^{*Y} \cdot_Y (a, a)$$

definiert die Baumsprache  $T = \{c(b^n a, b^n a) \mid n \in \mathbb{N}\}$ . Dies ist keine reguläre Baumsprache.

## 4.2 Asynchrone Baumautomaten

Im Folgenden wird ein Automatenmodell vorgestellt, welches die Klasse **Rat** der rationalen Baumrelationen definiert. Radmacher hat ein solches Modell basierend auf der Definition durch Multivariablen von Raoult entwickelt [28, 29].

Die grundlegende Idee ist, analog zu den Multivariablen in einem Automaten bestimmte Zustände so in Relation zueinander zu setzen, dass diese nur gemeinsam erreicht und verlassen werden können. Dies wird durch die Einführung von *Makrozuständen* erreicht.

**Definition 4.2.1.** Ein Makrozustand  $\mathfrak{q} = (q_1, \dots, q_m)$  ist eine nicht leere Folge von Zuständen. Die Menge aller Instanzen eines Makrozustandes ist das kartesische Produkt  $\mathfrak{q} \times \mathbb{N}$ . Alle Zustände eines Makrozustandes müssen vollständig und in gleicher Instanz auftreten. Instanznummern werden dabei eindeutig vergeben. Wir schreiben  $q_i \in \mathfrak{q} = (q_1, \dots, q_m)$  für  $q_i \in \{q_1, \dots, q_m\}$  und  $\mathfrak{q} \times \mathbb{N}$  für  $\{q_1, \dots, q_m\} \times \mathbb{N}$ .

Zunächst soll intuitiv die Funktionsweise eines entsprechenden Automaten erklärt werden: Ein  $n$ -asynchroner Baumautomat arbeitet auf  $n$ -Tupeln von beschränkt verzweigten Bäumen. Die Tupel werden bottom-up eingelesen. Der Automat hat eine Menge von paarweise verschiedenen Makrozuständen über einer endlichen Zustandsmenge. Jeder Zustand ist also einem Makrozustand eindeutig zugeordnet. Um genau Relationen der Klasse **Rat** zu erkennen, benötigt der Automat drei grundlegende Mechanismen:

**Synchrones Einlesen von Teilbäumen:** Alle Zustände eines Makrozustandes, die in gleicher Instanz auftreten, müssen in genau einem Berechnungsschritt des Automaten erreicht bzw. verlassen werden. Das impliziert, dass bestimmte Transitionen gleichzeitig ausgeführt werden.

**Asynchrones Einlesen von Teilbäumen:** Die Anzahl der Zustände eines Makrozustandes ist beliebig, daher müssen nicht alle Bäume eines  $n$ -Tupels in einem Berechnungsschritt berücksichtigt werden. Wenn der Automat beispielsweise nur einelementige Makrozustände enthält, findet keinerlei Synchronisation zwischen Eingebäumen statt. Zudem sind in der Transitionsrelation  $\varepsilon$ -Transitionen zugelassen, sodass Zustandswechsel ohne ein Fortschreiten auf der Eingabe möglich sind.

**Instantiierung von Zuständen:** Die Zustände des Automaten werden mit *Instanzvariablen* aus einer endlichen Variablenmenge kombiniert. Diese Variablen werden in einem Lauf des Automaten durch eine Instantiierungsfunktion auf natürliche Zahlen abgebildet. In einem Berechnungsschritt wird genau ein Makrozustand erreicht, dieser, und damit auch alle seine Zustände, werden mit einer Instanznummer versehen. Sind in einer Transition Zustände mit unterschiedlichen Instanzvariablen kombiniert, müssen sie auch unterschiedlich instantiiert sein, damit die Transition ausgeführt werden kann.

Um die asynchronen Baumautomaten formal definieren zu können, benötigen wir einige Begriffe und Schreibweisen. Für die Kombination eines Makrozustandes  $\mathbf{q}$  mit einer Instanzvariablen  $i$  schreiben wir  $(\mathbf{q}, i) = ((q_1, i), \dots, (q_m, i))$ . Die Menge aller Makrozustände eines Automaten ist mit  $\Omega$  bezeichnet und  $Var$  ist eine endliche Menge von Instanzvariablen. Eine *Instantiierungsfunktion*  $I : Var \rightarrow \mathbb{N}$  bildet eine Variable auf eine natürliche Zahl ab. Die Menge aller Knoten eines  $n$ -Tupels  $(t_1, \dots, t_n)$  von Bäumen ist mit  $dom_{(t_1, \dots, t_n)}$  bezeichnet.

**Definition 4.2.2.** Ein *Schnitt* eines  $n$ -Tupels  $(t_1, \dots, t_n)$  von Bäumen ist eine Menge von Knoten  $C \subseteq (dom_{(t_1, \dots, t_n)})$ , wobei für jeden Baum jeweils höchstens ein Knoten aus demselben Pfad von der Wurzel bis zu einem Blatt im Schnitt eines Tupels enthalten sein darf.

**Definition 4.2.3.** Ein  $n$ -asynchroner Baumautomat ist ein Tupel

$$\mathfrak{A}^{(n)} = (\Omega, Q, Var, \Sigma, \Delta, \mathcal{F})$$

mit einer endlichen Menge von disjunkten Makrozuständen  $\Omega$ , einer endlichen Menge von Zuständen  $Q$ , einem Rangalphabet  $\Sigma = \Sigma_0 \cup \dots \cup \Sigma_m$ , einer endlichen Variablenmenge  $Var$ , einer Menge von Endmakrozuständen  $\mathcal{F} \subseteq \Omega$  und einer Transitionsrelation

$$\begin{aligned} \Delta \subseteq & (Q \times Var) \times \{\varepsilon\} \times Q \times Var \\ & \cup \bigcup_{i=0}^m ((Q \times Var)^i \times \Sigma_i \times Q \times Var) . \end{aligned}$$

Die *Konfiguration* eines Automaten ist eine Abbildung  $c : C \rightarrow Q \times \mathbb{N}$ , die jedem Knoten aus dem Schnitt  $C$  einen Zustand und eine Instanzvariable zuordnet. Zuständen eines Makrozustandes wird durch eine Instantiierungsfunktion die gleiche Instanznummer zugewiesen; tritt ein Zustand innerhalb einer Konfiguration mehrmals auf, erhält er jeweils verschiedene Instanznummern:  $c(v_1) \neq c(v_2)$ .

Der Automat  $\mathfrak{A}$  führt einen Berechnungsschritt zwischen zwei Konfigurationen  $c_1 : C_1 \rightarrow Q \times \mathbb{N}$  und  $c_2 : C_2 \rightarrow Q \times \mathbb{N}$  durch, wenn die folgenden Voraussetzungen erfüllt sind:

- In  $C_2$  ist eine Menge von Knoten  $\{v_1, \dots, v_k\}$  enthalten, die die Elternknoten von Knoten  $\{v_{11}, \dots, v_{1r_1}, \dots, v_{k1}, \dots, v_{kr_k}\}$  aus  $C_1$  sind. Die Knoten aus  $C_2$  werden dabei durch Lesen ihrer Beschriftung durch geeignete Transitionen erreicht.
- In  $C_2$  gibt es eine Menge von Knoten  $\{v_{\varepsilon_1}, \dots, v_{\varepsilon_j}\}$ , für die  $\varepsilon$ -Transitionen benutzt werden.
- $C_2$  enthält alle Knoten aus  $C_1$ , für die in diesem Schritt keine Aktion durchgeführt wurde.

Die Zustände, die den Knoten  $v_{11}, \dots, v_{1r_1}, \dots, v_{k1}, \dots, v_{kr_k}, v_{\varepsilon_1}, \dots, v_{\varepsilon_j}$  zugewiesen werden, müssen dabei vollständige Makrozustände in jeweils gleicher Instanz bilden. Die Zustände der erreichten Knoten  $v_1, \dots, v_k, v_{\varepsilon_1}, \dots, v_{\varepsilon_j}$  bilden genau einen Makrozustand und sind alle mit der gleichen Instanznummer versehen. Die *Startkonfiguration* ist durch den leeren Schnitt  $C = \emptyset$  gegeben, in einem Konfigurationsübergang können dann nur Blätter von Bäumen eingelesen werden. Eine Konfiguration heißt *akzeptierend*, wenn die Wurzeln aller Eingabebäume auf genau einen Endmakrozustand abgebildet werden. Die *Relation* eines Automaten ist die Menge aller  $n$ -Tupel von Bäumen, die er erkennt.

Ein kurzes Beispiel soll die Funktionsweise dieser Automaten erläutern.

**Beispiel 4.2.1.** Ein 2-asynchroner Baumautomat sei gegeben durch  $\mathfrak{A}^{(2)} = (\mathfrak{Q}, Q, Var, \Sigma, \Delta, \mathcal{F})$  mit:

- $\mathfrak{Q} = \{(q_{a_1}, q_{a_2}), (q_{b_1}, q_{b_2}), (q_{c_1}, q_{c_2})\}$ ,
- $Q = \{q_{a_1}, q_{a_2}, q_{b_1}, q_{b_2}, q_{c_1}, q_{c_2}\}$ ,
- $Var = \{i, j\}$
- $\mathcal{F} = \{(q_{c_1}, q_{c_2})\}$  und

$$\Delta = \{ \begin{array}{l} (a, (q_{a_1}, i)), (a, (q_{a_2}, i)) \\ ((q_{a_1}, i)(q_{a_1}, j), c, (q_{c_1}, i)), ((q_{a_2}, i)(q_{a_2}, j), c, (q_{c_2}, i)), \\ ((q_{c_1}, i), \varepsilon, (q_{b_1}, i)), ((q_{c_2}, i), b, (q_{b_2}, i)), \\ ((q_{a_1}, i)(q_{b_1}, i), c, (q_{c_1}, i)), ((q_{a_2}, i)(q_{b_2}, i), c, (q_{c_2}, i)), \end{array} \}$$

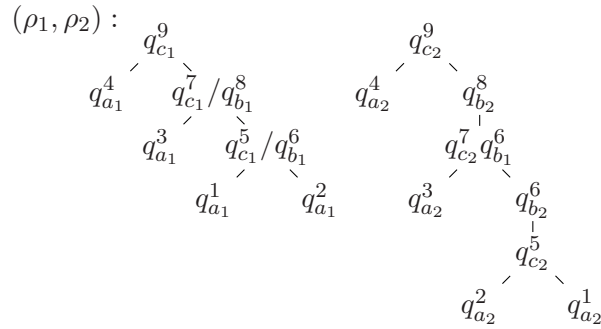


Abbildung 4.3: Lauf  $(\rho_1, \rho_2)$  des Automaten  $\mathfrak{A}^{(2)}$  auf einem Tupel  $(t_1, t_2) \in R$

Dieser Automat definiert die Relation  $R$  aus Beispiel 4.0.1. In Abbildung 4.3 ist ein Lauf  $(\rho_1, \rho_2)$  des Automaten  $\mathfrak{A}$  auf dem Tupel  $(t_1, t_2) \in R$  aus Abbildung 4.2 (iii) dargestellt.

Radmacher hat verschiedene Resultate für diese asynchronen Baumautomaten gezeigt, wir wollen einige hier kurz vorstellen.

**Satz 4.2.1** (Radmacher). *Eine  $n$ -stellige Relation  $R$  über Bäumen ist rational genau dann, wenn ein  $n$ -asynchroner Baumautomat  $\mathfrak{A}^{(n)}$  existiert mit  $\mathcal{R}(\mathfrak{A}^{(n)}) = R$ .*

Dieser Satz kann durch Induktion über den Aufbau der rationalen Baumrelationen nach Raoult gezeigt werden.

Die Unentscheidbarkeit einiger Entscheidungsprobleme kann sofort durch einen Bezug zu den rationalen Wortrelationen gezeigt werden. Dazu werden unäre Bäume betrachtet, Bäume nur mit Symbolen der Stelligkeit 1. Durch diese Bäume werden Worte kodiert und damit kann eine Äquivalenz zwischen rationalen Wortrelationen und rationalen Baumrelationen gezeigt werden. Nach Satz 2.2.2 folgt damit dann sofort:

**Korollar 4.2.1.** *Für zwei  $n$ -asynchrone Baumautomaten  $\mathfrak{A}_1^{(n)}$  und  $\mathfrak{A}_2^{(n)}$  ist es im Allgemeinen unentscheidbar, ob*

- $\mathcal{R}(\mathfrak{A}_1^{(n)}) \cap \mathcal{R}(\mathfrak{A}_2^{(n)}) = \emptyset$
- $\mathcal{R}(\mathfrak{A}_1^{(n)}) \subseteq \mathcal{R}(\mathfrak{A}_2^{(n)})$
- $\mathcal{R}(\mathfrak{A}_1^{(n)}) = \mathcal{R}(\mathfrak{A}_2^{(n)})$
- $\mathcal{R}(\mathfrak{A}_1^{(n)}) = \mathcal{T}_\Sigma \times \dots \times \mathcal{T}_\Sigma$ .

Die Entscheidbarkeiten des Nichtleerheitsproblems und das Unendlichkeitsproblems werden jeweils durch Angabe eines Algorithmus, der Mengen von erreichbaren Makrozuständen berechnet gezeigt:

**Satz 4.2.2** (Radmacher). *Für einen  $n$ -asynchronen Baumautomaten  $\mathfrak{A}^{(n)}$  ist es entscheidbar, ob*

- $R(\mathfrak{A}^{(n)}) = \emptyset$
- $R(\mathfrak{A}^{(n)})$  unendlich.

Eine interessante Eigenschaft und auch Schwäche der rationalen Baumrelationen ist, dass die einstelligen Relationen nicht den regulären Baumsprachen entsprechen. Betrachtet man dagegen für rationale Wortrelationen nach Definition 2.2.1 eine Relation  $R \subseteq \Sigma^*$ , so gibt es eine reguläre Sprache  $L$ , sodass

$$(w) \in R \Leftrightarrow w \in L .$$

Die einstelligen rationalen Wortrelationen entsprechen damit den regulären Wortsprachen. Da die Definition der rationalen Baumrelation eine Synchronisation von Teilbäumen ein und desselben Baumes erlaubt, ist diese Eigenschaft nicht übertragbar.

**Korollar 4.2.2.** *Die Klasse  $\text{Rat}_1$  der einstelligen rationalen Baumrelationen entspricht nicht der Klasse der regulären Baumsprachen.*



## Kapitel 5

# Rationale Relationen über unbeschränkt verzweigten Bäumen

In diesem Kapitel werden rationale Relationen über unbeschränkt verzweigten Bäumen definiert und analysiert. Diese Klasse von Relationen wird in der Folge mit  $\mathbf{uRat}$  bezeichnet; wenn explizit  $n$ -stellige Relationen gemeint sind, schreiben wir  $\mathbf{uRat}_n$ . Wir haben im vorherigen Kapitel drei äquivalente Ansätze für rationale Relationen über beschränkt verzweigten Bäumen kennen gelernt: rationale Ausdrücke mit Multivariablen, Multivariablengrammatiken und asynchrone Baumautomaten. Reguläre Sprachen von unbeschränkt verzweigten Bäumen sind durch unbeschränkt verzweigte Baumautomaten definiert, da beispielsweise DTDs von geringerer Aussagekraft sind [7]. Es liegt daher nahe, Relationen über unbeschränkt verzweigten Bäumen direkt durch ein Automatenmodell zu definieren und die anderen Ansätze zu vernachlässigen. Dies hat Radmacher bereits durch eine Erweiterung des asynchronen Baumautomaten aus Kapitel 4.2 getan. Der resultierende *asynchrone unbeschränkt verzweigte Baumautomat* wird hier vorgestellt und weiterführend analysiert. Dabei wird gezeigt, dass die für Rangalphabete betrachteten entscheidbaren Eigenschaften auch hier entscheidbar bleiben.

Wir werden sehen, dass mit diesen Automaten ein direkter Bezug zu rationalen Wortrelationen herzustellen ist, und dass die einstelligen Baumrelationen nicht den regulären Baumsprachen entsprechen.

Im Folgenden werden wir mit den *transitions-synchronisierten Baumautomaten* ein äquivalentes Automatenmodell definieren, dessen Idee die Synchronisation von Transitionen anstelle von Zuständen ist. Jede Transition wird eindeutig einem Baum aus einem Eingabetupel zugeordnet, dies erleichtert die Lesbarkeit, und wir werden es im nächsten Kapitel als Grundlage für Automaten auf Baumtupeln mit Anzahlbedingungen nutzen.

Im letzten Abschnitt schränken wir  $\mathbf{uRat}$  derart ein, dass einstellige Baumrelationen genau den regulären Baumsprachen entsprechen. Diese neue Klasse von *transitions-separierten Baumrelationen* wird mit  $\mathbf{SepuRat}$  bezeichnet. Einführend stellen wir ein Beispiel für eine unbeschränkt verzweigte Baumrelation vor.

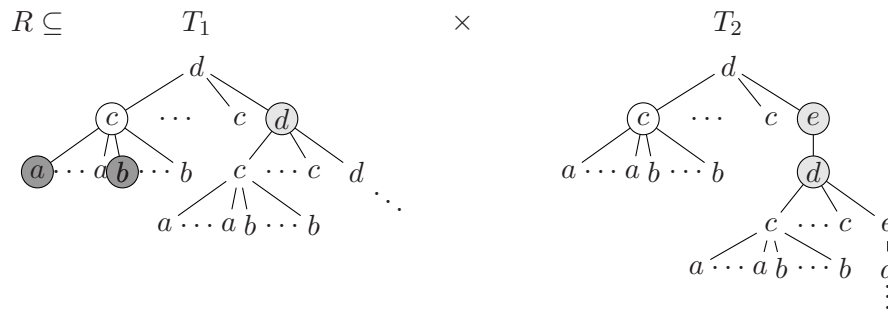


Abbildung 5.1: Relation von unbeschränkt verzweigten Baumsprachen  $T_1, T_2$

**Beispiel 5.0.2.** Seien  $T_1$  und  $T_2$  zwei unbeschränkt verzweigte Baumsprachen über dem Alphabet  $\Sigma = \{a, b, c, d\}$ . Beide sind in Abbildung 5.1 skizziert. Bäume aus  $T_1$  haben beliebige Höhe. Ihre Wurzel ist mit  $d$  beschriftet. Der linkeste Nachfolger ist mit  $c$  beschriftet und hat Blätter als Nachfolger, deren Beschriftungen als Wort von links nach rechts gelesen in der Sprache des regulären Ausdrucks  $a^*b^*$  sind. Rechts von dem mit  $c$  beschrifteten Nachfolger der Wurzel ist eine beliebige Anzahl von Blättern, die wiederum mit  $c$  beschriftet sind. Der rechteste Wurzelnachfolger ist mit  $d$  beschriftet, und seine Nachfolger genügen den gleichen Bedingungen wie die Wurzel. Die Sprache  $T_2$  ist ähnlich zu  $T_1$ , nur dass der rechteste Nachfolger der Wurzel mit  $e$  beschriftet ist und genau einen Nachfolger hat, der mit  $d$  beschriftet ist und den gleichen Bedingungen wie die Wurzel genügt. Die Relation  $R$  beinhaltet Tupel  $(t_1, t_2)$  mit  $t_1 \in T_1$ ,  $t_2 \in T_2$ , sodass für jedes  $c$  in  $t_1$  genau ein  $c$  in  $t_2$  auftritt. Für jedes  $d$  in  $t_1$  gibt es ein  $e$  und seinen Nachfolger  $d$  in  $t_2$ . Für beide Bäume gilt, dass es für jedes  $a$  genau ein  $b$  im selben Baum gibt. Die Anzahl an  $a$  und  $b$  in  $t_1$  kann unterschiedlich zu der in  $t_2$  sein.

Aus diesem Beispiel wird ersichtlich, dass auch die Klasse  $\mathbf{uRat}$  eine Synchronisation sowohl zwischen verschiedenen Bäumen eines Tupels von Bäumen als auch zwischen Teilbäumen eines einzigen Baumes beinhaltet.

## 5.1 Asynchrone unbeschränkt verzweigte Baumautomaten

Bislang wurden rationale Baumrelation nur für Rangalphabete definiert; die verschiedenen Ansätze sind in Kapitel 4 nachzulesen. Wir werden hier eine Erweiterung des asynchronen Baumautomaten nach Definition 4.2.3 vorstellen, die von Radmacher definiert wurde [28]. Zudem werden wir für dieses erweiterte Modell die Entscheidbarkeit

oder Unentscheidbarkeit verschiedener Probleme zeigen und die Beziehung zu rationalen Wortrelationen herausstellen.

Wie für Automaten auf unbeschränkt verzweigten Bäumen üblich, werden in den Transitionen reguläre Sprachen über der Zustandsmenge des Automaten betrachtet. Auch hier benötigen wir die Kombination der Zustände mit Instanzvariablen, um eine Synchronisation zwischen bestimmten Teilbäumen zu ermöglichen. Das führt zu der folgenden Transitionsrelation:

$$\Delta \subseteq \text{Reg}(Q \times \text{Var}) \times \Sigma \times (Q \times \text{Var}) .$$

Wir sprechen im Folgenden von der *Sprache der Transitionen*, wenn der in den Transitionen definierte reguläre Ausdruck aus  $\text{Reg}(Q \times \text{Var})$  gemeint ist.

Die signifikanten Eigenschaften eines asynchronen Automaten bleiben erhalten:

- Gleiche Zustände müssen innerhalb der gleichen Konfiguration durch unterschiedliche Instanznummern unterscheidbar sein, es dürfen also niemals zwei gleiche Zustände mit der gleichen Instanznummer kombiniert sein.
- Alle Zustände eines Makrozustandes treten komplett in gleicher Instanz auf.
- Sind in einer Transition Zustände mit unterschiedlichen Instanzvariablen kombiniert, müssen sie unterschiedlich instantiiert sein.
- Mit jedem Berechnungsschritt des Automaten wird genau ein Makrozustand erreicht.

Betrachtet man beispielsweise eine Transition  $((q_1, i)^*, a, (q_2, j))$ , so ist offensichtlich, dass schon bei zweimaligem Auftreten von  $q_1$  eine Verletzung der ersten Bedingung vorliegt, da der Variablen  $i$  hier durch eine Instantiierungsfunktion genau eine Instanznummer zugewiesen würde. Um die Definition nicht gültiger Transitionen zu verhindern, wird die Erweiterung  $\text{ext}(L)$  einer regulären Sprache  $L$  benutzt, die einem Symbol in seinem  $i$ -ten Auftreten in einem Wort den Index  $i$  zuordnet und es somit innerhalb des Wortes einzigartig werden lässt.

**Definition 5.1.1.** Sei  $w = (q_1, i) \dots (q_n, i)$  ein Wort über einem Alphabet  $Q \times \text{Var}$ . Die *Erweiterung*  $\text{ext}(w) = (q_1, i_1) \dots (q_n, i_n)$  des Wortes ist dadurch definiert, dass dem Symbol  $(q, i)$  in seinem  $j$ -ten Vorkommen in  $w$  der Index  $j$  zugeordnet wird:  $(q, i_j)$ . Die Erweiterung einer Sprache  $L \in \text{Reg}(Q \times \text{Var})$  ist definiert durch  $\text{ext}(L) = \{\text{ext}(w) \mid w \in L\}$ .

In diesem Kontext führt dies zu der Benutzung einer erweiterten Variablenmenge  $\text{Var} \times \mathbb{N}$ . Zur Erläuterung geben wir ein Beispiel für eine Transition an.

**Beispiel 5.1.1.** Sei die Transition eines asynchronen Automaten durch  $((q_1, i)^*, a, (q_2, j))$  gegeben. Nach der ursprünglichen Definition darf diese Transition nur für genau einen oder keinen Zustand  $(q_1, i)$  ausgeführt werden, der reguläre Ausdruck  $r = (q_1, i)^*$  wäre gleichbedeutend mit  $r = ((q_1, i) \vee \varepsilon)$ . Betrachtet man aber z.B. für ein Wort  $w = (q_1, i)(q_1, i)(q_1, i)(q_1, i)$ , das aus vorhandenen Zuständen für die Kinder eines Knotens gebildet werden kann, die Spracherweiterung, erhält man:

$$ext(w) = (q_1, i_1)(q_1, i_2)(q_1, i_3)(q_1, i_4) .$$

Somit sind alle Zustände eindeutig mit Variablen kombiniert und mit  $ext(w)$  kann der Zustand  $q_2$  erreicht werden.

Die Spracherweiterung wird für die asynchronen Baumatomen folgendermaßen genutzt. Zu einem Makrozustand  $q = (q_1, \dots, q_m)$  muss es Transitionen

$$(L_1, a_1, (q_1, i)), \dots, (L_m, a_m, (q_m, i))$$

geben mit  $L_i \in Reg(Q \times Var)$  und  $a_i \in \Sigma$  für  $1 \leq i \leq m$ . Ein Berechnungsschritt des Automaten kann ausgeführt werden, wenn

- es Sequenzen  $ext(w_1), \dots, ext(w_m) \in ext(Reg(Q \times Var))$  gibt mit  $ext(w_i) \in ext(L_i)$  und
- $ext(w_1) \cup \dots \cup ext(w_m)$  aus vollständigen, jeweils eindeutigen Instanzen von Makrozuständen besteht.

Die Verwendung der Spracherweiterung führt hinsichtlich der Funktionalität des Automaten zu keinen Einschränkungen, wie das folgende Lemma zeigt.

**Lemma 5.1.1** (Radmacher). *Für ein endliches Alphabet  $\Sigma$  ist das Wortproblem für die Spracherweiterung einer regulären Sprache entscheidbar. Darüber hinaus lässt sich die Spracherweiterung eines endlichen Wortes effektiv berechnen.*

Für die Definition der asynchronen Automaten über unbeschränkt verzweigten Bäumen werden einige Begriffe benötigt. Mit  $EVar := Var \times \mathbb{N}$  sei eine erweiterte Variablenmenge bezeichnet. Die *Instantiierung* einer Menge von Variablen  $\mathcal{V} \subseteq EVar$  ist eine Funktion  $I : \mathcal{V} \rightarrow \mathbb{N}, i_n \mapsto \alpha_n$ .  $\alpha$  ist die Instanznummer der Variablen  $i$ , und  $n \in \mathbb{N}$  ist die Zahl, die der Variablen  $i$  durch die Benutzung der Spracherweiterung zugeordnet wird. Der *Schnitt* eines  $n$ -Tupels wird wie in Definition 4.2.2 benutzt.

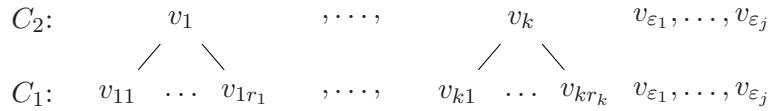


Abbildung 5.2: Schnitte  $C_1$  und  $C_2$  eines Konfigurationsüberganges

**Definition 5.1.2.** Ein  $n$ -asynchroner unbeschränkt verzweigter Baumautomat ist ein Tupel

$$\mathfrak{A}^{(n)} = (\mathfrak{Q}, Q, Var, \Sigma, \Delta, \mathcal{F})$$

mit einer endlichen Menge von disjunkten Makrozuständen  $\mathfrak{Q}$ , einer endlichen Menge von Zuständen  $Q$ , einem endlichen Alphabet  $\Sigma$ , einer endlichen Variablenmenge  $Var$ , einer Menge von Endmakrozuständen  $\mathcal{F} \subseteq \mathfrak{Q}$  und einer Transitionsrelation

$$\begin{aligned}
 \Delta \subseteq & \quad Reg(Q \times Var) \times \Sigma \times (Q \times Var) \\
 & \cup (Q \times Var) \times \{\varepsilon\} \times (Q \times Var) .
 \end{aligned}$$

Die *Konfiguration* eines Automaten ist eine Abbildung  $c : C \rightarrow Q \times \mathbb{N}$ , die jedem Knoten aus dem Schnitt  $C$  einen Zustand und eine Instanzvariable zuordnet. Zuständen eines Makrozustandes wird durch eine Instanziierungsfunktion die gleiche Instanznummer zugewiesen; tritt ein Zustand innerhalb einer Konfiguration mehrmals auf, erhält er jeweils verschiedene Instanznummern:  $c(v_1) \neq c(v_2)$ .

Der Automat  $\mathfrak{A}$  führt einen Berechnungsschritt zwischen zwei Konfigurationen  $c_1 : C_1 \rightarrow Q \times \mathbb{N}$  und  $c_2 : C_2 \rightarrow Q \times \mathbb{N}$  durch, wenn die folgenden Voraussetzungen erfüllt sind:

- In  $C_2$  ist eine Menge von Knoten  $\{v_1, \dots, v_k\}$  enthalten, die die Elternknoten von Knoten  $\{v_{11}, \dots, v_{1r_1}, \dots, v_{k1}, \dots, v_{kr_k}\}$  aus  $C_1$  sind. Die Knoten aus  $C_2$  werden dabei durch Lesen ihrer Beschriftung durch geeignete Transitionen erreicht.
- In  $C_2$  gibt es eine Menge von Knoten  $\{v_{\varepsilon_1}, \dots, v_{\varepsilon_j}\}$ , für die  $\varepsilon$ -Transitionen benutzt werden.
- $C_2$  enthält alle Knoten aus  $C_1$ , für die in diesem Schritt keine Aktion durchgeführt wurde.

Eine solche Situation ist in Abbildung 5.2 dargestellt. Die Bedingungen lauten formalisiert:

- $\{v_{11}, \dots, v_{kr_k}, v_{\varepsilon_1}, \dots, v_{\varepsilon_j}\} \subseteq C_1$
- $C_2 = (C_1 \setminus \{v_{11}, \dots, v_{kr_k}\}) \cup \{v_1, \dots, v_k\}$
- Es existieren geeignete Transitionen

$$(L_1, \text{val}(v_1), (q_1, h)), \dots, (L_k, \text{val}(v_k), (q_k, h))$$

und  $\varepsilon$ -Transitionen

$$((q_{\varepsilon_1}, h_1), \varepsilon, (q_{\varepsilon'_1}, h)), \dots, ((q_{\varepsilon_j}, h_j), \varepsilon, (q_{\varepsilon'_j}, h))$$

in  $\Delta$ , und für alle  $i \in \{1, \dots, k\}$  existiert ein Wort  $w_i$  über  $Q \times EVar$  mit

$$w_i = (q_{i1}, j_{i1}) \dots (q_{ir_i}, j_{ir_i}) \in \text{ext}(L_i)$$

und es existiert eine geeignete Instanziierung der Variablen aus  $EVar$ , sodass

- die Zustände  $q_{11}, \dots, q_{kr_k}, q_{\varepsilon_1}, \dots, q_{\varepsilon_j}$  genau eine Menge von vollständigen Makrozuständen bilden, die jeweils mit der gleichen Variablen kombiniert und gleich instanziiert sind und
- die durch Transitionen erreichten Zustände  $q_1, \dots, q_k, q_{\varepsilon'_1}, \dots, q_{\varepsilon'_j}$  einen vollständigen Makrozustand bilden, sodass alle Zustände mit der gleichen Variablen kombiniert und gleich instanziiert sind.

Eine *Startkonfiguration* bildet einen leeren Schnitt auf Zustände ab, nutzt also Transitionen aus  $\Sigma \times Q \times Var$ . Eine Konfiguration  $c : C \rightarrow Q \times \mathbb{N}$  ist *akzeptierend* genau dann, wenn

- $C$  die Menge der Wurzeln aller Eingabebäume  $(t_1, \dots, t_n)$  ist, und
- es einen Endmakrozustand  $\mathbf{q} = (q_1, \dots, q_n)$  gibt, sodass  $c$  die Wurzeln in gleicher Instanz auf die Zustände von  $\mathbf{q}$  abbildet.

Die Bedingung, dass jeder Zustand innerhalb einer Transition nur genau einmal in gleicher Instanz auftreten darf, ist notwendig, um eine korrekte Synchronisation zwischen den Bäumen eines Eingabetupels zu gewährleisten. Ebenso können Transitionen nur ausgeführt werden, wenn Makrozustände mit mehreren Zuständen vollständig und in gleicher Instanz auftreten. Im Falle der unbeschränkt verzweigten Bäume führt dies zu einer besonderen Art der Synchronisierung, die wir anhand eines Beispiels erläutern werden.

**Beispiel 5.1.2.** Seien  $\mathcal{T}_1$  und  $\mathcal{T}_2$  unbeschränkt verzweigte Baumsprachen.  $\mathcal{T}_1$  ist die Menge der Bäume der Höhe 2, deren Wurzel mit  $d$ , und deren Nachfolger alle mit  $c$  beschriftet sind.  $\mathcal{T}_2$  ist die Menge der Bäume der Höhe 2, deren Wurzel mit  $a$ , und deren Nachfolger ebenfalls mit  $c$  beschriftet sind. In Abbildung 5.3 sind unter (i) diese beiden Sprachen skizziert. Die Punkte zwischen den Baumknoten stellen eine beliebige Verzweigung dar.

Von Interesse sind die Eigenschaften von Relationen  $R \subseteq \mathcal{T}_1 \times \mathcal{T}_2$ , die durch asynchrone Baumautomaten über unbeschränkt verzweigten Bäumen definiert werden können.

Sei dazu  $\mathfrak{A}_1^{(2)} = (\mathfrak{Q}, Q, Var, \Sigma, \Delta, \mathcal{F})$  ein 2-asynchroner, unbeschränkt verzweigter Baumautomat mit

- $\mathfrak{Q} = \{(q_{c_1}), (q_{c_2}), (q_{d_1}, q_{a_2})\}$
- $Q = \{q_{c_1}, q_{c_2}, q_{d_1}, q_{a_2}\}$
- $Var = \{i\}$
- $\Sigma = \{a, c, d\}$
- $\mathcal{F} = \{(q_{d_1}, q_{a_2})\}$  und
- $\Delta = \{ (c, (q_{c_1}, i)), (c, (q_{c_2}, i)), \\ ((q_{c_1}, i)^*, d, (q_{d_1}, i)), ((q_{c_2}, i)^*, a, (q_{a_2}, i)) \\ \}$

In Abbildung 5.3 (ii) ist ein Lauf dieses Automaten auf einem Eingabetupel dargestellt, in dem der erste Baum genau  $m$  Knoten mit der Beschriftung  $c$ , und der zweite Baum genau  $n$  Knoten mit der Beschriftung  $c$  hat. Der Automat  $\mathfrak{A}_1^{(2)}$  akzeptiert ein Tupel  $(t_1, t_2)$ , wenn  $t_1 \in \mathcal{T}_1$  und  $t_2 \in \mathcal{T}_2$  gilt.

Betrachten wir nun einen Automaten  $\mathfrak{A}_2^{(2)}$ , der genau wie  $\mathfrak{A}_1^{(2)}$  definiert ist, nur dass die beiden Zustände  $q_{c_1}$  und  $q_{c_2}$  in  $\mathfrak{Q}$  zu einem Makrozustand  $(q_{c_1}, q_{c_2})$  zusammengefasst werden. Ein Lauf dieses Automaten ist in Abbildung 5.3 (iii) dargestellt, er kann nur von dieser Form sein. Da wir zwei Zustände zu einem Makrozustand zusammengefasst haben, können Transitionen für  $q_{c_1}$  und  $q_{c_2}$  nur noch simultan ausgeführt werden, für jeden Zustand  $q_{c_1}$  muss es genau einen Zustand  $q_{c_2}$  in exakt gleicher Instanz geben. Darum werden von diesem Automaten nur Tupel von Bäumen akzeptiert, in denen gleich viele Knoten mit  $c$  beschriftet sind.

Das Beispiel 5.1.2 zeigt, dass wir durch die asynchronen Automaten eine horizontale Synchronisation zwischen verschiedenen Bäumen definieren können. Es stellt sich nun

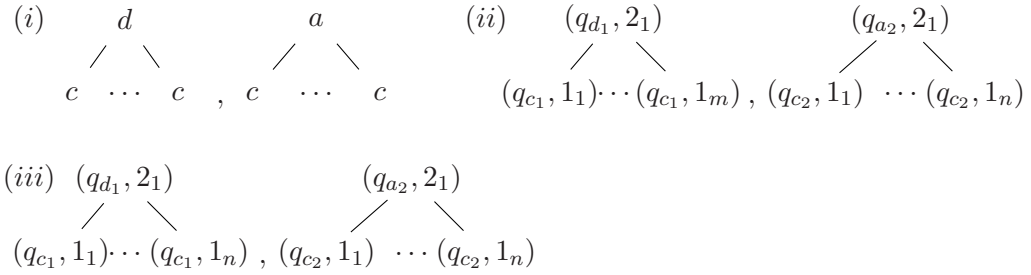


Abbildung 5.3: (i) Baumsprachen  $\mathcal{T}_1$  und  $\mathcal{T}_2$ , (ii) Lauf von  $\mathfrak{A}_1^{(2)}$ , (iii) Lauf von  $\mathfrak{A}_2^{(2)}$

die Frage, ob dieses Modell wegen dieser Eigenschaft zu mächtig für die Definition einer rationalen unbeschränkt verzweigten Baumrelation ist. Zum einen kann eine solche Synchronisation durch die Definition des Automaten aber verhindert werden, zum anderen ist dieser Automat eine natürliche Erweiterung des asynchronen Automaten für beschränkt verzweigte Bäume. Wir definieren daher die rationalen unbeschränkt verzweigten Baumrelationen durch diese Automaten.

**Definition 5.1.3.** Eine  $n$ -stellige Relation  $R \subseteq \mathcal{T}_1 \times \dots \times \mathcal{T}_n$  über unbeschränkt verzweigten Bäumen ist rational genau dann, wenn es einen  $n$ -asynchronen unbeschränkt verzweigten Baumautomaten  $\mathfrak{A}^{(n)}$  gibt, sodass

$$R = \mathcal{R}(\mathfrak{A}^{(n)}) .$$

Interessant ist hier auch die Betrachtung einstelliger Relationen. Im Falle der rationalen Wortrelationen entsprechen einstellige Relationen genau den regulären Sprachen von Wörtern. Wie in Kapitel 4.2 erwähnt, gilt dies für die rationalen Relationen über beschränkt verzweigten Bäumen nicht. Das folgende Beispiel zeigt, dass auch mit den asynchronen unbeschränkt verzweigten Baumautomaten nicht-reguläre Sprachen definierbar sind.

**Beispiel 5.1.3.** Sei  $\mathfrak{A}_{abc}^{(1)} = (\mathfrak{Q}, Q, Var, \Sigma, \Delta, \mathcal{F})$  ein 1-asynchroner, unbeschränkt verzweigter Baumautomat mit

- $\mathfrak{Q} = \{(q_b, q_c), (q_a)\}$
- $Q = \{q_a, q_b, q_c\}$

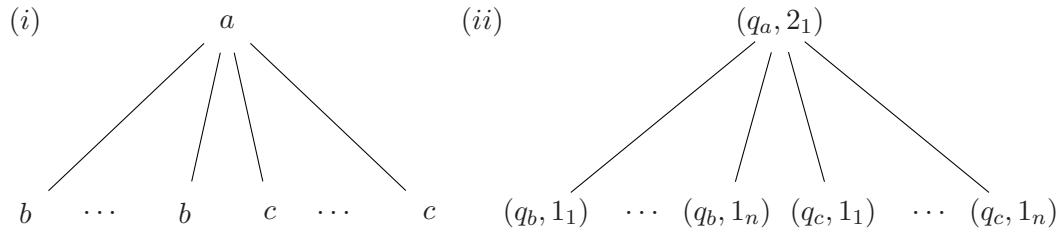


Abbildung 5.4: (i) Baumsprache  $\mathcal{T}_{abc}$ , (ii) Lauf des Automaten  $\mathfrak{A}_{abc}^{(1)}$

- $Var = \{i, j\}$
- $\Sigma = \{a, b, c\}$
- $\mathcal{F} = \{(q_a)\}$
- $\Delta = \{ (b, (q_b, i)), (c, (q_c, i)), \\ ((q_b, i)^*(q_c, i)^*, a, (q_a, j)) \\ \}$

Dieser Automat erkennt genau die Sprache  $\mathcal{T}_{abc}$ : Alle Bäume haben die Höhe 2, ihre Wurzel ist mit  $a$  beschriftet. Die Beschriftungen der Kinder der Wurzel bilden von links nach rechts gelesen ein Wort, das in der Sprache des regulären Ausdruckes  $r = b^*c^*$  ist. Dadurch, dass  $q_b$  und  $q_c$  in einem Makrozustand zusammengefasst sind, müssen sie in gleicher Anzahl und in jeweils gleicher Instanz auftreten. Es gilt also für alle Bäume  $t \in \mathcal{T}_{abc}$ , dass  $t_{|a|} = t_{|c|}$ . Die Baumsprache und ein Lauf des Automaten sind in Abbildung 5.4 dargestellt.

Durch die Definition des Automaten enthält die Transition  $((q_b, i)^*(q_c, i)^*, a, (q_a, j))$  semantisch einen Ausdruck der Form  $r = b^nc^n$ . Dies ist keine reguläre Sprache, damit ist  $\mathcal{T}_{abc}$  auch keine reguläre Baumsprache. Aus diesem Beispiel schließen wir das folgende Korollar.

**Korollar 5.1.1.** *Die Klasse der einstelligen unbeschränkt verzweigten Baumrelationen entspricht nicht der Klasse der regulären unbeschränkt verzweigten Baumsprachen.*

### 5.1.1 Beziehung zu rationalen Wortrelationen

Um eine genauere Beschreibung der Klasse  $\text{uRat}$  zu erhalten und einige unentscheidbare Eigenschaften dieser durch asynchrone Automaten definierten Relationen zu folgern, soll ein Bezug zu rationalen Wortrelationen hergestellt werden. Für ein Wort  $u = a_1 a_2 \dots a_n$  betrachten wir dazu den unären Baum  $u\# = a_1(a_2(\dots(a_n(\#))\dots))$ , dessen einziges Blatt mit  $\#$  beschriftet ist. Auf diese Weise kann zu einer Wortrelation eine zugehörige Baumrelation definiert werden.

**Definition 5.1.4.** Sei  $R \subseteq \Sigma_1^* \times \dots \times \Sigma_n^*$  eine  $n$ -stellige Wortrelation mit  $\# \notin \Sigma_i, 1 \leq i \leq n$ . Dann ist die zu  $R$  gehörige Baumrelation  $\mathcal{U}(R)$  über den Beschriftungsalphabeten  $\Sigma_1 \cup \{\#\}, \dots, \Sigma_n \cup \{\#\}$  definiert durch

$$\mathcal{U}(R) = \{(u_1\#, \dots, u_n\#) \mid (u_1, \dots, u_n) \in R\}$$

Wünschenswert ist eine Äquivalenz zwischen diesen beiden Relationen, wir werden dazu zeigen, dass eine Wortrelation genau dann rational ist, wenn die zugehörige Baumrelation  $\mathcal{U}(R)$  rational ist, also von einem unbeschränkt verzweigten asynchronen Baumautomaten erkannt wird.

**Lemma 5.1.2.** Sei  $R$  eine rationale Wortrelation. Dann ist die zu  $R$  gehörige Baumrelation  $\mathcal{U}(R)$  rational.

*Beweis.* Sei  $R$  gegeben durch einen asynchronen Automaten  $\mathfrak{A} = (Q_1, \Sigma_1, \dots, \Sigma_n, q_0, \Delta_1, F_1)$  mit  $\mathcal{R}(\mathfrak{A}) = R$ . Sei  $Q_1 = \{q_0, \dots, q_m\}$ . Wir konstruieren dazu einen  $n$ -asynchronen unbeschränkt verzweigten Baumautomaten  $\mathfrak{A}^{(n)} = (\mathfrak{Q}_2, \mathfrak{Q}_2, \text{Var}, \Sigma_1 \cup \dots \cup \Sigma_n \cup \{\#\}, \Delta_2, \mathcal{F})$  mit

- $Q_2 = \{q_j^1, \dots, q_j^n \mid q_j \in Q_1, 1 \leq j \leq m\} \cup \{q_f^1, \dots, q_f^n\}$
- $\mathfrak{Q}_2 = \{(q_0^1, \dots, q_0^n), \dots, (q_m^1, \dots, q_m^n)\} \cup \{(q_f^1, \dots, q_f^n)\}$
- $\text{Var} = \{i\}$
- $\Delta_2 = \begin{aligned} & \{(\#, (q_j^1, i)), \dots, (\#, (q_j^n, i)) \mid q_j \in F_1\} \\ & \cup \{((q_k^1, i), a_1, (q_j^1, i)), \dots, ((q_k^n, i), a_n, (q_j^n, i)) \mid (q_j, a_1 / \dots / a_n, q_k) \in \Delta_1\} \\ & \cup \{((q_0^1, i), \varepsilon, (q_f^1, i)), \dots, ((q_0^n, i), \varepsilon, (q_f^n, i))\} \\ & \text{mit } a_i \in \Sigma_i \cup \{\varepsilon\}, 1 \leq j, k \leq m. \end{aligned}$
- $\mathcal{F} = \{(q_f^1, \dots, q_f^n)\}$

Durch diese Konstruktion erhält man einen Automaten  $\mathfrak{A}^{(n)}$ , sodass gilt:  $(u_1, \dots, u_n) \in R \Leftrightarrow (u_1, \dots, u_n) \in \mathcal{R}(\mathfrak{A}) \Leftrightarrow (u_1\#, \dots, u_n\#) \in \mathcal{R}(\mathfrak{A}^{(n)})$ . Also ist nach Definition 5.1.4  $\mathcal{R}(\mathfrak{A}^{(n)}) = \mathcal{U}(R)$ . Damit ist  $\mathcal{U}(R)$  rational.  $\square$

Auch die andere Richtung dieser Beziehung kann gezeigt werden; der Konstruktion ist aufgrund der Synchronisation durch Makrozustände wesentlich komplizierter.

**Lemma 5.1.3.** *Sei  $R$  eine Wortrelation. Ist die zugehörige Baumrelation  $\mathcal{U}(R)$  rational, so ist  $R$  rational.*

*Beweis.* Sei die Wortrelation  $R$  gegeben durch einen  $n$ -asynchronen unbeschränkt verzweigten Baumautomaten  $\mathfrak{A}^{(n)} = (\mathfrak{Q}_1, Q_1, Var, \Sigma_1 \cup \dots \cup \Sigma_n \cup \{\#\}, \Delta_1, \mathcal{F}_1)$  mit  $\mathcal{R}(\mathfrak{A}^{(n)}) = \mathcal{U}(R)$ . Die Transitionen des Automaten liegen o.B.d.A. nur mit atomaren regulären Ausdrücken in den Transitionen vor, zudem vernachlässigen wir das Symbol  $\#$ .

$$\Delta_1 \subseteq Q \times Var \times \Sigma \times \mathfrak{Q} \times Var .$$

Wir konstruieren einen asynchronen Automaten  $\mathcal{A} = (Q_2, \Sigma_1, \dots, \Sigma_n, q_0, \Delta_2, F_2)$ . Durch die Transitionsrelation  $\Delta_2$  müssen die möglicherweise mehr elementigen Makrozustände aus  $\mathfrak{A}^{(n)}$  berücksichtigt werden, daher definieren wir die Zustände des asynchronen Automaten als Tupel von Zuständen:

- $Q_2 \subseteq \bigcup_{n \in \mathbb{N}} Q^n \cup \{q_0\}$
- $q_0 \in Q_2$

Die folgende Konstruktion der Menge wird für alle Endmakrozustände aus  $\mathfrak{q}_f \in \mathcal{F}$  durchgeführt:

- Betrachte für Endmakrozustand  $\mathfrak{q}_f = (q_1, \dots, q_n)$  mit  $1 \leq j \leq o$  alle Transitionen

$$\delta_i = ((q'_i, l'), a_i, (q_i, l)) \in Q \times Var \times \Sigma \times \{q_i\} \times Var, \quad 1 \leq i \leq n$$

aus  $\Delta_1$ . Für Transitionen  $\delta_1 \dots \delta_n$  bilden die entsprechenden Zustände  $q'_1, \dots, q'_n$  nach Definition des asynchronen Baumautomaten vollständige Makrozustände  $\mathfrak{q}_1, \dots, \mathfrak{q}_m$  mit  $1 \leq m \leq n$ .

- Markiere für alle möglichen Kombinationen von Transitionen  $\delta_1, \dots, \delta_n$  jeden der Zustände  $q'_1, \dots, q'_n$  mit dem Makrozustand  $\mathfrak{q}_j$ , zu dem er gehört für  $1 \leq j \leq m$ . Es resultiert eine Folge von 2-Tupeln von Zuständen:

$$(q'_1, \mathfrak{q}'_1), \dots, (q'_n, \mathfrak{q}'_n)$$

mit  $q'_j \in \{q_1, \dots, q_m\}$  und  $\bigcup_{i=j}^n \mathfrak{q}_j = \{q_1, \dots, q_m\}$  für  $1 \leq j \leq n$ .

- Betrachte für  $\delta_1, \dots, \delta_n$  die Folge von Symbolen  $(a_1, \dots, a_n)$ , wobei  $a_i$  durch  $\delta_i = ((q'_i, l'), a_i, (q_i, l))$  bestimmt ist. Bilde für jeden Makrozustand  $\mathfrak{q}_j \in \{\mathfrak{q}_1, \dots, \mathfrak{q}_m\}$  eine Folge  $(a'_1, \dots, a'_n)$  mit  $a'_i \in \Sigma \cup \{\varepsilon\}$ :

$$a'_i = \begin{cases} a_i, & \text{wenn } \mathfrak{q}'_i = \mathfrak{q}_j \\ \varepsilon & \text{sonst} \end{cases}$$

- Bilde für jeden Makrozustand  $\mathfrak{q}_j = (q_1, \dots, q_o)$  und die Transitionen  $\delta_1, \dots, \delta_o \in \Delta_1$  mit  $\delta_i = ((q'_i, l'), a_i, (q_i, l)), 1 \leq i \leq o$  eine Transition

$$\delta_j = ((q_1, \dots, q_o) \times a'_1 / \dots / a'_n \times (q'_1 \dots q'_o))$$

und füge  $(q_1, \dots, q_o)$  und  $(q'_1 \dots q'_o)$  zu  $Q_2$  und  $\delta_j$  zu  $\Delta_2$  hinzu.

Diese Konstruktion ist iterativ für die neuen Zustände, die Makrozuständen entsprechen, analog durchzuführen. Endzustände sind diejenigen Zustandstupel, die als Makrozustände im ursprünglichen Automaten durch das Lesen von Blättern erreicht werden. Startzustand ist  $q_0$ , und es wird für jeden Endmakrozustände  $\mathfrak{q}_f$  eine Transition

$$(q_0, \varepsilon_1 / \dots / \varepsilon_n, \mathfrak{q})$$

zu  $\Delta_2$  hinzugefügt.

Mit dieser Konstruktion gilt  $(u_1, \dots, u_n) \in R \Leftrightarrow (u_1\#, \dots, u_n\#) \in \mathcal{R}(\mathfrak{A}^{(n)}) \Leftrightarrow (u_1, \dots, u_n) \in \mathcal{R}(\mathfrak{A})$ .  $R$  wird also von einem asynchronen Automaten erkannt und ist damit rational.  $\square$

Aus Lemma 5.1.2 und Lemma 5.1.3 folgt direkt die Gültigkeit des folgenden Satzes:

**Satz 5.1.1.** *Eine Wortrelation  $R$  ist genau dann rational, wenn die zugehörige Baumrelation  $\mathcal{U}(R)$  rational ist.*

Wir haben gezeigt, dass die hier definierten Relationen aus  $\text{uRat}$  eine natürliche Erweiterung der rationalen Wortrelationen sind, im folgenden Kapitel können wir daher alle Unentscheidbarkeiten direkt übertragen.

### 5.1.2 Entscheidungsprobleme

**Korollar 5.1.2.** *Für zwei  $n$ -asynchrone unbeschränkt verzweigte Baumautomaten  $\mathfrak{A}_1^{(n)}$  und  $\mathfrak{A}_2^{(n)}$  ist es im Allgemeinen unentscheidbar, ob*

- $R(\mathfrak{A}_1^{(n)}) \cap R(\mathfrak{A}_2^{(n)}) = \emptyset$

- $R(\mathfrak{A}_1^{(n)}) \subseteq R(\mathfrak{A}_2^{(n)})$
- $R(\mathfrak{A}_1^{(n)}) = R(\mathfrak{A}_2^{(n)})$
- $R(\mathfrak{A}_1^{(n)}) = \mathcal{T}_\Sigma \times \dots \times \mathcal{T}_\Sigma$  .

Die Entscheidbarkeit des Nichtleerheitsproblems bleibt auch für Baumrelationen erhalten. Wir werden dazu einen Algorithmus angeben, der Erreichbarkeitsmengen für Makrozustände berechnet. Dazu muss zunächst festgelegt werden, wann eine Menge von Transitionen ausgeführt werden kann. Ist in einer Transition eine Sprache über Zuständen angegeben, in deren Wörtern mindestens ein Zustand Element eines mehr-elementigen Makrozustandes ist, so müssen die anderen Zustände des Makrozustandes kombiniert mit derselben Variable jeweils in der Sprache dieser oder einer anderen Transition enthalten sein. Um einen Makrozustand  $\mathfrak{q} = (q_1, \dots, q_m)$  eines Automaten  $\mathfrak{A}^{(n)} = (\Omega, Q, Var, \Sigma, \Delta, \mathcal{F})$  ausgehend von bisher erreichten Makrozuständen zu erreichen,

- muss eine Variable  $i \in Var$  existieren und für alle Zustände  $q_j \in \mathfrak{q}$  muss es Transitionen  $(L_j, a_j, (q_j, i))$  geben mit  $L_j \in Reg(Q \times Var)$ ,  $a_j \in \Sigma$ ,  $1 \leq j \leq m$ , und
- für alle bisher erreichten Makrozustände  $\mathfrak{q}$  mit  $|\mathfrak{q}| > 1$  muss für alle  $q \in \mathfrak{q}$  die Anzahl aller  $(q, h) \in Q \times Var$ , die benötigt werden, um Wörter aus den  $L_j$  zu bilden, jeweils genau gleich sein.

Durch die erste Bedingung werden Transitionen ausgewählt, mit denen die Zustände eines Makrozustandes erreicht werden können. Die zweite Bedingung gewährleistet die korrekte Benutzung der schon vorher erreichten Makrozustände: Es werden nur Transitionen ausgeführt, sodass Wörter aus den Sprachen der Transitionen nur aus Paaren  $(q, h)$  gebildet werden, die wiederum komplette Makrozustände in jeweils gleicher Instanz bilden. Wir drücken diese zweite Bedingung durch eine Presburger-Formel  $\varphi$  aus, die aussagt, dass für alle Makrozustände die in ihnen enthaltenen Zustände kombiniert mit der gleichen Variablen in gleicher Anzahl auftreten. Dazu werden freie Variablen  $y_{(q,i)}$  für alle Paare von Zuständen und Instanzvariablen aus der Menge  $Y_{Q \times Var} = \{y_{(q,i)} \mid (q, i) \in Q \times Var\}$  benutzt. Die Presburger-Formel lautet:

$$\varphi = \bigwedge_{(q_1, \dots, q_m) \in \Omega} \bigwedge_{h \in Var} (y_{(q_1, h)} = \dots = y_{(q_m, h)}) .$$

Diese Formel werden wir für den Beweis des folgenden Satzes nutzen.

**Satz 5.1.2.** *Das Nichtleerheitsproblem für asynchrone unbeschränkt verzweigte Baumautomaten ist entscheidbar.*

*Beweis.* Wir geben einen Algorithmus an, der die Menge aller erreichbaren Makrozustände  $E_{\mathfrak{A}}$  berechnet. Sei dazu  $\mathfrak{A}^{(n)} = (\mathfrak{Q}, Q, Var, \Sigma, \Delta, \mathcal{F})$  ein  $n$ -asynchroner unbeschränkt verzweigter Baumautomat.  $E_k$  sei die Menge der Makrozustände, die nach  $k$  Iterationen des Algorithmus hinzugefügt wurden.  $Q_{E_k}$  sei die Menge aller Zustände, die in Makrozuständen aus  $E_k$  enthalten sind, also  $Q_{E_k} = \bigcup_{\mathfrak{q} \in E_k} \mathfrak{q}$ .

Zunächst wird der Algorithmus angegeben, anschließend wird genau definiert, wie die Mengen  $E_k$  gebildet werden.

$E_0 := \emptyset$

$k := 0$

**REPEAT**

$k := k + 1$

$E_k := E_{k-1} \cup \{\mathfrak{q}\}$ , mit „ $\mathfrak{q}$  ist mit Makrozuständen aus  $E_{k-1}$  erreichbar.“

**UNTIL**  $E_k = E_{k-1}$

**IF**  $E_k \cap \mathcal{F} = \emptyset$  **THEN** RETURN " leer "

**ELSE** RETURN " nicht leer "

In jedem Schritt des Algorithmus wird also genau ein Makrozustand hinzugefügt; ist dies nicht so, ist  $E_k$  die vollständige Menge  $E_{\mathfrak{A}}$  von erreichbaren Makrozuständen. Der Algorithmus terminiert mit der Überprüfung, ob ein Endmakrozustand erreichbar ist oder nicht und entsprechender Ausgabe.

Die Menge  $E_k := E_{k-1} \cup \{\mathfrak{q}\}$  mit  $\mathfrak{q} = (q_1, \dots, q_m) \in \mathfrak{Q}, m \geq 1$ , wird in der  $k$ -ten Iteration des Algorithmus folgendermaßen gebildet:

1. Für alle Zustände  $q_j \in \mathfrak{q}$  mit  $1 \leq j \leq m$  existiert eine Variable  $i \in Var$  sodass Transitionen  $(L_j, a_j, (q_j, i)) \in \Delta$  existieren mit  $L_j \in Reg(Q \times Var)$  und  $a_j \in \Sigma \cup \{\varepsilon\}$ . Alle diese Transitionen werden einer Menge  $\Delta' \subseteq \Delta$  hinzugefügt.
2. Für jeden Zustand  $q_j$  muss es für mindestens eine Transition  $(L_j, a_j, (q_j, i)) \in \Delta'$  ein Wort über  $Q_{E_{k-1}} \times Var$  geben, das in der Sprache des regulären Ausdrucks  $L_j$  ist. Dieses Problem ist nach Lemma 2.1.1 entscheidbar. Alle möglichen Transitionen werden einer Menge  $\Delta''$  hinzugefügt.
3. Sei  $(L_1, a_1, (q_1, i)), \dots, (L_m, a_m, (q_m, i))$  eine nach 1. und 2. mögliche Kombination von Transitionen aus  $\Delta''$ 
  - Für jeden bisher erreichten Makrozustand  $\mathfrak{q} \in E_{k-1}$  mit  $|\mathfrak{q}| > 1$  und jede Variable  $i \in Var$  wird eine Presburger-Formel  $\lambda_{\mathfrak{q}}$  nach Satz 3.2.3 berechnet, die das Parikh-Bild der Sprache  $L_1 \cdot L_2 \cdot \dots \cdot L_m$  über dem eingeschränkten Alphabet  $\mathfrak{q} \times Var$  beschreibt.

- Für jeden der Makrozustände  $\mathfrak{q} \in E_{k-1}$  wird eine Formel  $\delta_{\mathfrak{q}}$  gebildet:

$$\delta_{\mathfrak{q}} = \lambda_{\mathfrak{q}} \wedge \varphi$$

mit

$$\varphi = \bigwedge_{(q_1, \dots, q_n) \in \Omega} \bigwedge_{i \in \text{Var}} (y_{(q_1, i)} = \dots = y_{(q_n, i)}), n \in \mathbb{N}.$$

- Alle  $\delta_{\mathfrak{q}}$  müssen erfüllbar sein. Dies ist nach Satz 3.1.2 entscheidbar.

Für die Blätter der Bäume können die Transitionen auch von der Form  $(a_i, \mathfrak{q})$  sein. Die Sprache  $L_j$  enthält dann nur das leere Wort.

Im 1. Schritt werden alle Transitionen ausgewählt, mit denen überhaupt die Zustände  $q_1, \dots, q_m$  des Makrozustandes erreicht werden können. Im 2. Schritt werden die Transitionen ausgeschlossen, für deren Sprachen mit bisher erreichten Zuständen keine Wörter gebildet werden können. Im 3. Schritt wird überprüft, ob eine möglichen Kombination von Transitionen ausgeführt werden kann, wenn die Bedingung gestellt wird, dass mit der gleichen Variablen kombinierte Zustände eines Makrozustandes in gleicher Anzahl in den Wörtern auftreten. Dabei wird das Parikh-Bild der Konkatenation berechnet, weil unabhängig von der resultierenden Sprache so Wörter aus jeder der einzelnen Sprachen betrachtet werden.

Der Algorithmus terminiert, da  $\Omega$  nur endlich viele Makrozustände enthält, damit werden höchstens  $|\Omega|$  viele Iterationen durchgeführt. Auf eine genaue Komplexitätsanalyse wird verzichtet, es ist aber zu bedenken, dass nach Kapitel ?? die Komplexität der Entscheidbarkeit von Presburger-Formeln schon LINA-TIME  $2^{2^{O(n)}}$  ist.  $\square$

Um die Funktionsweise des Nichtleerheitstests zu verdeutlichen, geben wir ein kurzes Beispiel an.

**Beispiel 5.1.4.** Seien  $\mathcal{T}_1$  und  $\mathcal{T}_2$  die Baumsprachen aus Beispiel 5.1.2. Sei analog zu den Automaten dieses Beispiels  $\mathfrak{A}_3^{(2)} = (\Omega, Q, \text{Var}, \Sigma, \Delta, \mathcal{F})$  ein 2-asynchroner, unbeschränkt verzweigter Baumautomat mit

- $\Omega = \{(q_{c_1}, q_{c_2}), (q_{d_1}, q_{a_2})\}$
- $Q = \{q_{c_1}, q_{c_2}, q_{d_1}, q_{a_2}\}$
- $\text{Var} = \{i, j\}$
- $\Sigma = \{a, c, d\}$
- $\mathcal{F} = \{(q_{d_1}, q_{a_2})\}$

- $\Delta = \{\delta_1, \delta_2, \delta_3, \delta_4\}$  mit
  - $\delta_1 = (c, (q_{c_1}, i)), \quad \delta_2 = (c, (q_{c_2}, i)),$
  - $\delta_3 = (((q_{c_1}, i)(q_{c_1}, i))^*, d, (q_{d_1}, j)), \quad \delta_4 = ((q_{c_2}, i)((q_{c_2}, i)(q_{c_2}, i))^*, a, (q_{a_2}, j))$

Die Transitionen  $\delta_3, \delta_4$  sind so definiert, dass Zustände  $q_{c_1}$  in gerader und Zustände  $q_{c_2}$  in ungerader Anzahl auftreten müssen. Es ist also nicht möglich,  $\delta_3$  und  $\delta_4$  so auszuführen, dass die Zustände  $q_{c_1}$  und  $q_{c_2}$  kombiniert mit der Variablen  $i$  in gleicher Anzahl genutzt werden. Diese beiden Zustände sind aber Elemente desselben Makrozustandes  $(q_{c_1}, q_{c_2})$  und man kann nur mit  $\delta_3, \delta_4$  einen Endmakrozustand erreichen. Die Sprache des Automaten  $\mathfrak{A}_3^{(2)}$  ist also leer. Der Algorithmus bestätigt dies nach 2 Iterationsschritten:

$k = 1$ : Um den Makrozustand  $(q_{c_1}, q_{c_2})$  hinzuzufügen, werden die Transitionen  $\delta_1, \delta_2$  ausgewählt. In Schritt 2 werden diese beiden Transitionen bestätigt, da trivialerweise mit der leeren Menge  $Q_{E_0}$  das leere Wort zu bilden ist. Im 3. Schritt werden die Transitionen endgültig ausgewählt, da keine Makrozustände in der Menge  $E_0$  sind und damit auch gar keine Formeln gebildet werden.  $E_1 = \{(q_{c_1}, q_{c_2})\}$ .

$k = 2$ : Um  $(q_{d_1}, q_{a_2})$  zu erreichen, werden  $\delta_3, \delta_4$  ausgewählt. In Schritt 2. werden diese Transitionen bestätigt, da über der Menge  $Q_{E_1} = \{q_{c_1}, q_{c_2}\}$  jeweils Wörter aus den Sprachen der beiden Transitionen gebildet werden können. In Schritt 3. werden für den einzigen bislang erreichten Makrozustand  $(q_{c_1}, q_{c_2})$  mit  $L_3 = ((q_{c_1}, i)(q_{c_1}, i))^*$  und  $L_4 = (q_{c_2}, i)((q_{c_2}, i)(q_{c_2}, i))^*$  die folgenden Formeln gebildet:

- $\lambda_{(q_{c_1}, q_{c_2})} = \exists y((2 \cdot y = y_{(q_{c_1}, i)}) \wedge (2 \cdot y + 1 = y_{(q_{c_2}, i)}))$
- $\varphi = (y_{(q_{c_1}, i)} = y_{(q_{c_2}, i)}) \wedge (y_{(q_{c_1}, j)} = y_{(q_{c_2}, j)}) \wedge (y_{(q_{d_1}, i)} = y_{(q_{a_2}, i)}) \wedge (y_{(q_{d_1}, j)} = y_{(q_{a_2}, j)})$
- $\varphi_{(q_{c_1}, q_{c_2})} = \lambda_{(q_{c_1}, q_{c_2})} \wedge \varphi$

Die Formel  $\lambda_{(q_{c_1}, q_{c_2})}$  sagt aus, dass  $(q_{c_1}, i)$  in gerader und  $(q_{c_2}, i)$  in ungerader Anzahl auftreten muss.  $\varphi$  sagt aus, dass genau diese Zustände zusammen mit der Variablen  $i$  in gleicher Anzahl auftreten müssen. Die Konjunktion dieser beiden Formeln ist unerfüllbar.  $(q_{c_1}, q_{c_2})$  wird nicht hinzugefügt, und der Algorithmus terminiert mit der Ausgabe, dass die Relation des Automaten leer ist.

## 5.2 Synchronisierte Transitionen

In diesem Kapitel wird ein alternatives Automatenmodell vorgestellt, das die Klasse `uRat` definiert. Die Idee ist es, anstatt Zustände des Automaten zu Makrozuständen zusammenzufassen, Tupel von Transitionen zu verwenden. Die Synchronisation zwischen

Elementen eines Eingabetupels von Bäumen erfolgt dadurch, dass die Transitionen eines Tupels in einem Berechnungsschritt des Automaten gleichzeitig ausgeführt werden müssen. In diesem Modell soll jede Transition eindeutig einem Baum eines Eingabetupels zugeordnet werden können, daher hat ein Automat, der  $n$ -Tupel von Bäumen als Eingabe hat, ausschließlich  $n$ -Tupel von Transitionen, die jeweils zusammen ausgeführt werden müssen. Wir nennen diese Transitionen *synchronisierte Transitionen*.

Das Modell wird äquivalent zu den asynchronen unbeschränkt verzweigten Baumautomaten definiert, sodass die bereits bewiesenen Eigenschaften und insbesondere die Definition der Klasse `uRat` durch diese Automaten direkt übernommen werden können. Die einzelnen Transitionen sind von der Form

$$Reg(Q \times Var) \times \Sigma \times Q \times Var .$$

Das Modell muss Teilbäume sowohl *synchron* als auch *asynchron* einlesen können. Die Realisierung wird zunächst erläutert.

**Synchrones Einlesen von Teilbäumen** Alle Transitionen eines Transitionstupels werden zusammen ausgeführt; es wird in jedem Schritt des Automaten durch ein solches Transitionstupel für jeden Baum der Eingabe eine Aktion definiert. Auch bei diesem Automatenmodell werden Instanznummern an Zustände vergeben, zu diesem Zwecke werden Zustände mit Instanzvariablen kombiniert. In einem Berechnungsschritt werden mit allen Transitionen eines Transitionstupels Zustände zusammen mit der gleichen Instanzvariablen erreicht, die eine Instantiierungsfunktion im  $i$ -ten Berechnungsschritt des Automaten auf die Instanznummer  $i$  abbildet. Es ist zu beachten, dass entsprechend den asynchronen Automaten auch eine Synchronisation innerhalb ein und desselben Eingabebaumes stattfinden kann, daher gibt es für einzelne Elemente eines  $n$ -Tupels von Bäumen *innere Tupel* von Transitionen. Ein Transitionstupel ist ein  $n$ -Tupel von Transitionstupeln, die beliebige aber endliche Stelligkeit haben.

**Asynchrones Einlesen von Teilbäumen** Innerhalb eines Transitionstupels ist es möglich, durch  $\varepsilon$ -Transitionen Zustände zu wechseln, ohne auf der Eingabe fortzuschreiten. Soll kein Zustandswechsel durchgeführt werden, wird an der entsprechenden Stelle im Transitionstupel vereinfachend ein  $\$$  eingefügt. Somit wird für bestimmte Bäume keine Aktion durchgeführt.

Ein kurzes Beispiel wird die Funktionsweise und die Analogie zu den asynchronen Baumautomaten verdeutlichen. Wir grenzen die inneren, auch einstelligen, Transitionstupel für genau einen Baum eines Eingabetupels durch eckige Klammern ein; äußere runde

Klammern umfassen das gesamte  $n$ -Tupel von Transitionen:  $([\dots], \dots, [\dots])$ . Leere innere Tupel werden wie schon erwähnt durch ein  $\$$  gekennzeichnet.

**Beispiel 5.2.1.** Ein 2-asynchroner unbeschränkt verzweigter Baumautomat  $\mathfrak{A}^{(2)}$  habe einen Makrozustand  $\mathfrak{q} = (q_1, q_2)$  und Transitionen  $(L_1, a, (q_1, i)), (L_2, a, (q_2, i))$ ,  $L_i \in \text{Reg}(Q \times \text{Var})$ . Ein äquivalenter transitions-synchronisierter Automat  $\mathfrak{B}^{(2)}$  benötigt Zustände  $q_1, q_2$  und ein 2-Tupel von Transitionen  $([(L_1, a, (q_1, i))], [(L_2, a, (q_2, i))])$ . Definiert der Makrozustand die Synchronisation innerhalb ein und desselben Baumes, benötigt man ein inneres Tupel von Transitionen:  $([(L_1, a, (q, i)), (L_2, a, (q, i))], \$)$ . Die beiden Transitionen werden auf den ersten Baum des Eingabetupels angewendet, für den zweiten wird keine Aktion ausgeführt. Es sei angemerkt, dass, wenn im weiteren Lauf keine Unterscheidung mehr notwendig ist, auch ein Zustand für den transitions-synchronisierten Automaten ausreicht.

Durch die eindeutige Instantiierung aller in einem Berechnungsschritt erreichten Zustände kann man rein intuitiv auf die Benutzung der Spracherweiterung, wie sie für asynchrone Baumautomaten benutzt wird, verzichten. Die eindeutige Zuordnung von in einem Berechnungsschritt erreichten Zuständen zueinander ist gegeben, auch die Bedingung, dass diese Zustände gemeinsam erreicht und verlassen werden, kann erfüllt werden. Um aber die Äquivalenz zu asynchronen Baumautomaten zu erreichen, kann auf die Einbindung der Spracherweiterung nicht verzichtet werden. Beide Ansätze erweitern die rationalen Wortrelationen auf natürliche Weise, doch um hier eine eindeutige Definition der Klasse  $\text{uRat}$  zu erhalten, wählen wir die äquivalente Definition.

Um den Automaten formal zu definieren, reicht der Schnitt eines  $n$ -Tupels von Bäumen nach Definition 4.2.2 nicht aus, es muss jeder Baum berücksichtigt werden:

**Definition 5.2.1.** Ein *vollständiger Schnitt* eines  $n$ -Tupels  $(t_1, \dots, t_n)$  von Bäumen ist ein Schnitt  $\mathcal{C} \subseteq (\text{dom}_{t_1, \dots, t_n})$  dieses Tupels und enthält genau einen Knoten eines jeden Pfades von einer Wurzel zu einem Blatt der Bäume  $t_1, \dots, t_n$ .

Die Instantiierungsfunktion  $I : \mathcal{V} \rightarrow \mathbb{N}^2$  vergibt in Berechnungsschritt  $i$  des Automaten die Instanznummer  $i$  an Variablen aus  $E\text{Var}$ . Wir nutzen die Menge  $E\text{Var}$  auch hier zusammen mit der Spracherweiterung nach Definition 5.1.1.

**Definition 5.2.2.** Ein transitions-synchronisierter Baumautomat auf  $n$ -Tupeln von unbeschränkt verzweigten Bäumen ist ein Tupel  $\mathfrak{A}^{(n)} = (Q, \text{Var}, \Sigma, \Delta, F)$  mit:

- $Q$  ist eine endliche Menge von Zuständen;
- $\text{Var}$  ist eine endliche Menge von Instanzvariablen;

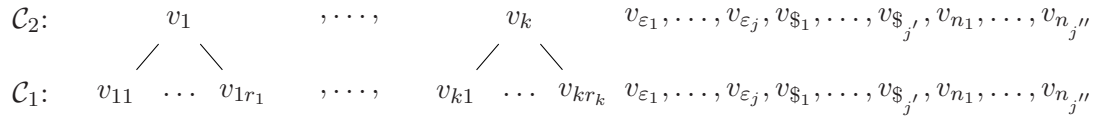


Abbildung 5.5: Vollständige Schnitte  $C_1$  und  $C_2$  eines Konfigurationsüberganges

- $\Sigma$  ist ein Beschriftungsalphabet;
- $F \subseteq Q$  ist eine Menge von Endzuständen;
- $\Delta$  ist eine Transitionsrelation mit

$$\Delta \subseteq \left( \left[ \bigcup_{r \in \mathbb{N}^+} ((\text{Reg}(Q \times \text{Var}) \times \Sigma \times Q \times \text{Var}) \cup (Q \times \text{Var} \times \{\varepsilon\} \times Q \times \text{Var}))^r \right] \cup \{\$\} \right)^n.$$

Eine *Konfiguration*  $c$  ist gegeben durch  $c : \mathcal{C} \rightarrow Q \times \text{Var}$ , sodass jedem Knoten aus einem vollständigen Schnitt  $\mathcal{C}$  ein Zustand und eine Instanzvariable zugeordnet sind. Variablen werden durch die Instantiierungsfunktion  $I$  auf eine natürliche Zahl abgebildet.

Der Automat  $\mathfrak{A}$  führt einen Berechnungsschritt auf einem Tupel  $(t_1, \dots, t_n)$  von unbeschränkt verzweigten Bäumen zwischen zwei Konfigurationen  $c : \mathcal{C}_1 \rightarrow Q \times \text{Var}$  und  $d : \mathcal{C}_2 \rightarrow Q \times \text{Var}$  durch, wenn es ein geeignetes Transitionstupel  $\delta^{(n)} \in \Delta$  gibt und gilt:

- $\mathcal{C}_2$  enthält Knoten  $v_1, \dots, v_k$ , die Elternknoten von Knoten  $v_{11}, \dots, v_{kr_k}$  aus  $\mathcal{C}_1$  sind. Die  $v_1, \dots, v_k$  werden durch Ausführen normaler Transitionen in Abhängigkeit von ihren Kindern erreicht.
- $\mathcal{C}_1$  und  $\mathcal{C}_2$  enthalten Knoten  $v_{\varepsilon_1}, \dots, v_{\varepsilon_j}$ , auf die  $\varepsilon$ -Transitionen mit Zustandswechsel angewendet werden.
- $\mathcal{C}_1$  und  $\mathcal{C}_2$  enthalten Knoten  $v_{\$_1}, \dots, v_{\$_j}$  aus Bäumen, für die in  $\delta^{(n)}$   $\$$ -Transitionen definiert sind.
- $\mathcal{C}_1$  und  $\mathcal{C}_2$  enthalten Knoten  $v_{n_1}, \dots, v_{n_j}''$ , für die keine  $\$$ -Transitionen definiert sind und keine Aktion durchgeführt werden soll.

Die beiden vollständigen Schnitte  $\mathcal{C}_1$  und  $\mathcal{C}_2$  sind in Abbildung 5.5 dargestellt. Die Bedingungen lauten formalisiert:

- $\mathcal{C}_1 = \{v_{11}, \dots, v_{kr_k}, v_{\varepsilon_1}, \dots, v_{\varepsilon_j}, v_{\$_1}, \dots, v_{\$_j}, v_{n_1}, \dots, v_{n_j}''\}$

- $C_2 = (C_1 \setminus \{v_{11}, \dots, v_{kr_k}\}) \cup \{v_1, \dots, v_k\}$
- In  $\delta^{(n)} = ([\delta_{11}, \dots, \delta_{1r_1}], \dots, [\delta_{n1}, \dots, \delta_{nr_n}]) \in \Delta, r_i \geq 1, 1 \leq i \leq n$  gibt es
  - $k$  Transitionen  $(L_1, \text{val}(v_1), (q_1, o)), \dots, (L_k, \text{val}(v_k), (q_k, o))$  aus  $\delta^{(n)}$  mit  $L_i \in \text{Reg}(Q \times \text{Var}), 1 \leq i \leq k, o \in \text{Var}, q_1, \dots, q_k \in Q$  und
  - $j$   $\varepsilon$ -Transitionen  $((q_{\varepsilon_1}, h_1), \varepsilon, (q_{\varepsilon'_1}, o)), \dots, ((q_{\varepsilon_j}, h_j), \varepsilon, (q_{\varepsilon'_j}, o))$  aus  $\delta^{(n)}$  und
  - $m' \leq m$   $\$$ -Transitionen aus  $\delta^{(n)}$ , die keine Aktion ausführen,
 und es gilt  $k, j, m \geq 0, k+j \geq 1$ . Damit wird mindestens ein neuer Zustand erreicht.

Die Konfiguration  $c$  bildet die Knoten  $v_{11}, \dots, v_{kr_k}$  auf Zustände  $q_{11}, \dots, q_{kr_k}$  ab. Diese Zustände bilden zusammen mit ihren zugehörigen Variablen für alle  $i \in (1, \dots, k)$  geeignete Wörter  $w_i$  über  $Q \times \text{EVar}$  mit

$$w_i = (q_{i1}, o_{i1}) \dots (q_{ir_i}, o_{ir_i}) \in \text{ext}(L_i)$$

und es existiert eine geeignete Instantiierung der Variablen aus  $\text{EVar}$ , sodass alle Knoten aus  $C_1$ , die auf Zustände abgebildet werden, welche mit derselben Instanznummer kombiniert sind, entweder vollständig durch Transitionen verlassen werden oder nicht. Sei dazu für  $n \in \mathbb{N}$ :

$$A_n = \{v \in \text{dom}_{t_1, \dots, t_n} \mid c_n : v \mapsto (q, n), q \in Q \text{ beliebig}\}$$

und es gilt für alle  $n$ :

$$A_n \subseteq \{v_{11}, \dots, v_{kr_k}, v_{\varepsilon_1}, \dots, v_{\varepsilon_j}\} \Leftrightarrow A_n \cap \{v_{\$1}, \dots, v_{\$j}, v_{n_1}, \dots, v_{n_j''}\} = \emptyset.$$

In einer *Startkonfiguration*  $c_0 : \emptyset \rightarrow Q \times \text{Var}$  wird der leere vollständige Schnitt auf Zustände mit Instanzvariablen abgebildet, es wird also ein Transitionstupel mit Transitionen nur aus  $\Sigma \times Q \times \text{Var}$  genutzt. Eine Konfiguration  $c : \mathcal{C} \rightarrow Q \times \mathbb{N}$  ist *akzeptierend*, genau dann, wenn

- $\mathcal{C}$  die Menge der Wurzeln aller Eingabebäume  $(t_1, \dots, t_n)$  ist, und
- es ein Transitionstupel  $\delta^{(n)}$  ohne  $\$$ -Transitionen gibt, sodass mit genau diesem Tupel ein Übergang in Konfiguration  $c$  möglich ist, und alle Knoten aus  $\mathcal{C}$  in gleicher Instanz ausschließlich auf einen akzeptierenden Zustand abgebildet werden.

Ein *Lauf* ist eine Folge von Konfigurationen, die durch ein Tupel von Bäumen dargestellt wird. Die Bäume dieses Tupels sind mit Zuständen des Automaten und den zugehörigen Instanznummern beschriftet. Die Bäume werden mit Symbolen des Alphabetes  $Q \times \text{Var}$  beschriftet. Ein *akzeptierender Lauf* endet mit einer akzeptierenden Konfiguration.

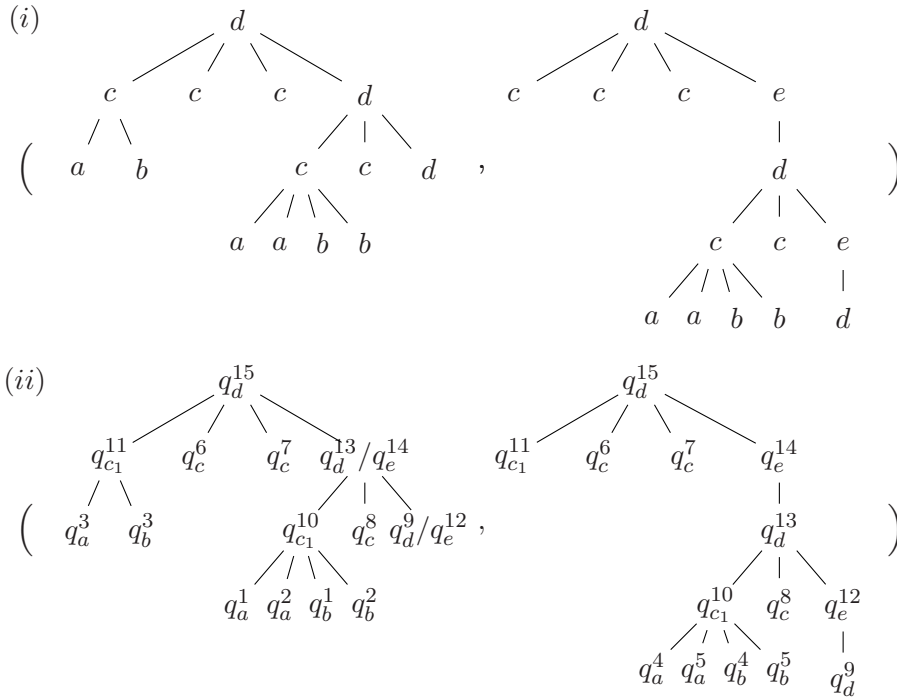


Abbildung 5.6: (i) Tupel  $(t_1, t_2) \in R$ , (ii) Lauf von  $\mathfrak{A}_1^{(n)}$  auf  $(t_1, t_2)$

Wir geben nun ein vollständiges Beispiel für einen solchen Automaten und seinen Lauf auf einem Eingabetupel an.

**Beispiel 5.2.2.** Sei  $\mathfrak{A}_1^{(n)} = (Q, Var, \Sigma, \Delta, F)$  ein transitions-synchronisierter Baumautomat auf 2-Tupeln von Bäumen mit:

- $Q = \{q_a, q_b, q_c, q_{c_1}, q_d, q_e\}$
- $Var = \{i\}$
- $\Sigma = \{a, b, c, d, e\}$
- $F = \{q_d\}$
- $\Delta = \{ ( [ (a, (q_a, i)), (b, (q_b, i)) ] , \$ ), ( \$ , [ (a, (q_a, i)), (b, (q_b, i)) ] ) \}$

$$\begin{aligned}
& ( [ ((q_a, i)^*(q_b, i)^*, c, (q_{c_1}, i)) ] , [ ((q_a, i)^*(q_b, i)^*, c, (q_{c_1}, i)) ] ), \\
& ( [ (c, (q_c, i)) ] , [ (c, (q_c, i)) ] ), \\
& ( [ (d, (q_d, i)) ] , [ (d, (q_d, i)) ] ), \\
& ( [ ((q_d, i), \varepsilon, (q_e, i)) ] , [ ((q_d, i), e, (q_e, i)) ] ), \\
& ( [ ((q_{c_1}, i)(q_c, i)^*(q_e, i), d, (q_d, i)) ] , [ ((q_{c_1}, i)(q_c, i)^*(q_e, i), d, (q_d, i)) ] ) \\
& \}
\end{aligned}$$

Der Automat  $\mathfrak{A}_1^{(n)}$  definiert die unbeschränkt verzweigte Baumrelation  $R$  aus Beispiel 5.0.2. In Abbildung 5.6 ist  $(i)$  ein Tupel  $(t_1, t_2) \in R$  dargestellt sowie  $(ii)$  ein akzeptierender Lauf von  $\mathfrak{A}_1^{(n)}$  auf diesem Tupel.

### 5.2.1 Äquivalenz zu asynchronen Baumautomaten

In diesem Abschnitt wird die Äquivalenz der asynchronen unbeschränkt verzweigten Baumautomaten nach Definition 5.1.2 und den hier vorgestellten transitions-synchronisierten Baumautomaten gezeigt. Die Äquivalenz impliziert, dass wir mit diesen Automaten einen weiteren Formalismus zur Definition der Klasse **uRat** von unbeschränkt verzweigten Baumrelationen bereitstellen.

In dem hier vorgestellten Beweis wird der Nichtdeterminismus beider Automatenmodelle genutzt, um für einen gegebenen Automaten des einen Modells jeweils einen des anderen zu konstruieren, der genau die gleiche Relation erkennt. Wir werden die *Menge aller Permutationen* einer gegebenen Menge benötigen, die wir zunächst definieren.

**Definition 5.2.3.** Die *Menge aller Permutationen ohne Wiederholungen* einer  $n$ -elementigen Menge  $A_n = \{a_1, \dots, a_n\}$  ist die Menge aller  $n$ -Tupel, in denen jedes Element aus  $A_n$  genau einmal auftritt. Diese Menge ist mit  $\mathcal{S}_{|A_n|}$  bezeichnet.

**Satz 5.2.1.** *Asynchrone unbeschränkt verzweigte Baumautomaten und transitions-synchronisierte Baumautomaten sind gleich mächtig.*

*Beweis.* Wir werden beide Richtungen des Beweises zeigen.

**Lemma 5.2.1.** *Für jeden asynchronen unbeschränkt verzweigten Baumautomaten  $\mathfrak{A}^{(n)}$  gibt es einen transitions-synchronisierten Baumautomaten  $\mathfrak{B}^{(n)}$ , sodass*

$$\mathcal{R}(\mathfrak{A}^{(n)}) = \mathcal{R}(\mathfrak{B}^{(n)}) .$$

*Beweis.* Sei ein asynchroner Baumautomat gegeben durch  $\mathfrak{A}^{(n)} = (\mathfrak{Q}_1, Q_1, Var_1, \Sigma_1, \Delta_1, \mathcal{F}_1)$ . Wir konstruieren dazu einen transitions-synchronisierten Baumautomaten  $\mathfrak{B}^{(n)} = (Q_2, Var_2, \Sigma_2, \Delta_2, F_2)$  mit

- $Q_2 = Q_1$
- $Var_2 = Var_1$
- $\Sigma_2 = \Sigma_1$
- $F_2 = \{q \mid q \in \mathfrak{q} \wedge \mathfrak{q} \in \mathcal{F}_1\}$

Die Menge  $\Delta_2$  von Transitionstupeln muss analog zu den Makrozuständen aus  $\mathfrak{Q}_1$  konstruiert werden, da durch die Transitionstupel die Synchronisation zwischen verschiedenen Elementen der Eingabe stattfindet.

Um die Struktur eines Eingabetupels zu determinieren, wird zunächst für alle Endmakrozustände  $\mathfrak{q}_f \in \mathcal{F}_1$  mit  $\mathfrak{q}_f = (q_1, \dots, q_n)$  die Menge  $\Delta_{\mathfrak{q}_f}$  aller Mengen von Transitionen  $\{\delta_1, \dots, \delta_n\}$  berechnet, mit denen die Zustände  $q_1, \dots, q_n$  erreicht werden können:

$$\Delta_{\mathfrak{q}_f} = \left\{ \left\{ \delta_1, \dots, \delta_n \right\} \mid \begin{array}{l} \delta_i \in Reg(Q_1 \times Var_1) \times \Sigma_1 \times \{q_i\} \times Var_1, \\ 1 \leq i \leq n \end{array} \right\}$$

Mit allen Transitionen müssen die neuen Endzustände  $\{q_1, \dots, q_n\}$  zusammen mit derselben Instanzvariablen erreicht werden, daher werden alle Transitionen so abgeändert, dass für ein festes  $j \in Var_2$  gilt:

$$\Delta'_{\mathfrak{q}_f} = \left\{ \left\{ \delta_1, \dots, \delta_n \right\} \mid \begin{array}{l} \delta_i \in Reg(Q_1 \times Var_1) \times \Sigma_1 \times \{q_i\} \times \{j\}, \\ 1 \leq i \leq n \end{array} \right\}$$

Die Reihenfolge dieser Transitionen in einem Transitionstupel ist durch den Endmakrozustand zwingend festgelegt, daher wird für jede Menge  $\{\delta_1, \dots, \delta_n\} \in \Delta'_{\mathfrak{q}_f}$  eine Menge  $\Delta_{\mathfrak{q}_f, \{\delta_1, \dots, \delta_n\}}$  gebildet, die alle gültigen  $n$ -Tupel von Transitionen  $\delta_1, \dots, \delta_n$  enthält.

$$\Delta_{\mathfrak{q}_f, \{\delta_1, \dots, \delta_n\}} = \left\{ (\delta_1, \dots, \delta_n) \mid \{\delta_1, \dots, \delta_n\} \in \Delta'_{\mathfrak{q}_f} \right\}.$$

Damit sind alle Transitionstupel, mit denen akzeptierende Konfigurationen erreicht werden können, analog zu den Endmakrozuständen konstruiert.

Für jeden Makrozustand  $\mathfrak{q} = (q_1, \dots, q_m) \in \mathfrak{Q}_1 \setminus \mathcal{F}_1$  mit  $m \in \mathbb{N}$  müssen alle Transitionen aus  $\Delta_1$ , mit denen die  $q_j \in \mathfrak{q}$  erreicht werden können, so zu  $m$ -elementigen Mengen zusammengefasst werden, dass mit den Transitionen einer Menge jeweils genau die Zustände  $q_1, \dots, q_m$  kombiniert mit derselben Instanzvariablen erreicht werden können. Aus

diesen Tupeln werden alle möglichen Permutationen von inneren und äußeren Transitionen zusammen mit der nötigen Anzahl an  $\$$ -Transitionen gebildet, dadurch stehen alle Möglichkeiten für einen Transitionsübergang zur Verfügung. Der resultierende Automat “rät” nichtdeterministisch die richtigen Transitionstupel und kann nur eine akzeptierende Konfiguration entsprechend den  $n$ -elementigen Endmakrozuständen erreichen. Zunächst wird für jeden Makrozustand  $\mathbf{q} = (q_1, \dots, q_m)$  die Menge  $\Delta_{\mathbf{q}}$  aller Mengen  $\{\delta_1, \dots, \delta_m\}$  von Transitionen berechnet, mit denen jeweils genau die Zustände  $q_1, \dots, q_m$  erreicht werden können:

$$\Delta_{\mathbf{q}} = \left\{ \left\{ \delta_1, \dots, \delta_m \right\} \mid \begin{array}{l} \delta_i \in \text{Reg}(Q_1 \times \text{Var}_1) \times \Sigma_1 \times \{q_i\} \times \text{Var}_1, \\ 1 \leq i \leq m \end{array} \right\}$$

Mit allen Transitionen müssen die Zustände  $\{q_1, \dots, q_n\}$  zusammen mit derselben Instanzvariablen erreicht werden, daher werden alle Transitionen so abgeändert, dass für ein festes  $j \in \text{Var}_2$  gilt:

$$\Delta'_{\mathbf{q}} = \left\{ \left\{ \delta_1, \dots, \delta_m \right\} \mid \begin{array}{l} \delta_i \in \text{Reg}(Q_1 \times \text{Var}_1) \times \Sigma_1 \times \{q_i\} \times \{j\}, \\ 1 \leq i \leq m \end{array} \right\}$$

Für jede Menge  $\{\delta_1, \dots, \delta_m\} \in \Delta'_{\mathbf{q}}$  wird eine Menge  $\Delta_{\mathbf{q}, \{\delta_1, \dots, \delta_m\}}$  gebildet, die alle gültigen  $n$ -Tupel von Transitionstupeln mit den Transitionen  $\delta_1, \dots, \delta_m$  enthält.

$$\Delta_{\mathbf{q}, \{\delta_1, \dots, \delta_m\}} = \left\{ (s_1, \dots, s_n) \mid \begin{array}{l} s_i \in \mathcal{S}_{|p_i|}, \\ p_i \in \mathcal{P}(\{\delta_1, \dots, \delta_m\}) \cup \{\{\$\}\}, \\ \bigcup_i (p_i) \setminus \{\$\} = \{\delta_1, \dots, \delta_m\}, \\ p_i \cap p_j = \emptyset, \\ 1 \leq i, j \leq n \end{array} \right\}.$$

Ein Tupelelement  $s_i$  ist eine Permutation der Menge  $p_i$ , also wieder ein Tupel. Die  $p_i$  sind disjunkte Teilmengen der Menge  $\{\delta_1, \dots, \delta_n\}$  oder  $p_i = \{\$\}$ . Die Vereinigung der  $p_i$  ohne  $\{\$\}$  entspricht genau der Menge  $\{\delta_1, \dots, \delta_n\}$ . Jedes  $\delta_i$  tritt in einem Tupel  $(s_1, \dots, s_n)$  genau einmal auf.

Die Menge  $\Delta_2$  ist die Vereinigung aller Mengen  $\Delta_{\mathbf{q},\{\delta_1,\dots,\delta_m\}}$  und aller Mengen  $\Delta_{\mathbf{q}_f,\{\delta_1,\dots,\delta_n\}}$ :

$$\begin{aligned} \Delta_2 := & \bigcup_{\mathbf{q} \in \mathfrak{Q}_1} \bigcup_{\{\delta_1,\dots,\delta_m\} \in \Delta'_q} \Delta_{\mathbf{q},\{\delta_1,\dots,\delta_m\}} \\ & \cup \bigcup_{\mathbf{q} \in \mathfrak{F}_1} \bigcup_{\{\delta_1,\dots,\delta_n\} \in \Delta'_{q_f}} \Delta_{\mathbf{q}_f,\{\delta_1,\dots,\delta_n\}} \end{aligned}$$

Damit ist die Konstruktion abgeschlossen. Es kann gezeigt werden, dass jedes Eingabetupel  $t^{(n)} = (t_1, \dots, t_n)$ , das von einem asynchronen Baumautomaten  $\mathfrak{A}$  akzeptiert wird, genau von dem hier konstruierten transitions-synchronisierten Baumautomaten  $\mathfrak{B}$  akzeptiert wird. Dazu muss der Lauf von  $\mathfrak{A}$  auf  $t^{(n)}$  mit  $\mathfrak{B}$  simuliert werden.  $\mathfrak{A}$  erreicht in jedem Berechnungsschritt genau einen Makrozustand in gleicher Instanz mit einer bestimmten Menge von Transitionen. Da für  $\mathfrak{B}$  in  $\Delta_2$  für jeden Makrozustand des Automaten  $\mathfrak{A}$  alle möglichen Kombinationen von Transitionstupeln, mit deren Transitionen genau die Zustände des Makrozustandes erreichbar sind, zur Verfügung stehen, kann jeder mögliche Berechnungsschritt von  $\mathfrak{A}$  auch für  $\mathfrak{B}$  ausgeführt werden. Ein Lauf von  $\mathfrak{A}$  ist akzeptierend, wenn er in einer akzeptierenden Konfiguration endet, die Wurzeln der Bäume also in gleicher Instanz auf genau einen Endmakrozustand abgebildet werden. In  $\mathfrak{B}$  werden dann genau die Transitionen, mit denen die Endmakrozustände erreicht werden, angewandt, und zwar in einem Transitionstupel, das genau die Zustände des entsprechenden Makrozustandes aus  $\mathfrak{A}$  erreicht.

$\mathfrak{B}$  akzeptiert damit genau die Tupel von Bäumen, die von  $\mathfrak{A}$  akzeptiert wird.  $\square$

**Lemma 5.2.2.** *Für jeden transitions-synchronisierten Baumautomaten  $\mathfrak{A}^{(n)}$  gibt es einen asynchronen unbeschränkt verzweigten Baumautomaten  $\mathfrak{B}^{(n)}$ , sodass  $\mathcal{R}(\mathfrak{A}^{(n)}) = \mathcal{R}(\mathfrak{B}^{(n)})$ .*

*Beweis.* Sei ein transitions-synchronisierter Baumautomat gegeben durch  $\mathfrak{A}^{(n)} = (Q_1, Var_1, \Sigma_1, \Delta_1, F_1)$ . Wir konstruieren dazu einen asynchronen unbeschränkt verzweigten Baumautomaten  $\mathfrak{B}^{(n)} = (\mathfrak{Q}_2, Q_2, Var_2, \Sigma_2, \Delta_2, \mathcal{F}_2)$  mit

- $Var_2 = Var_1$
- $\Sigma_2 = \Sigma_1$

Die Makrozustände aus  $\mathfrak{Q}_2$  müssen dabei analog zu den Transitionstupeln des Automaten  $\mathfrak{A}$  gebildet werden; die Transitionen aus  $\Delta_2$  entsprechen denen der Tupel aus  $\Delta_1$ . Dabei müssen zunächst die Transitionstupel beachtet werden, mit denen eine akzeptierende Konfiguration erreichbar ist, da diese genau die  $n$ -elementigen Endmakrozustände

des asynchronen Baumautomaten festlegen. Sei dazu  $\Delta_f \subseteq \Delta_1$  die Menge solcher Transitionstupel.

$$\Delta_f = \{([\delta_1], \dots, [\delta_n]) \in \Delta \mid \forall i : \\ \delta_i \in (Reg(Q \times Var) \times \Sigma \times F \times Var) \\ \cup Q \times Var \times \{\varepsilon\} \times F \times Var, \\ 1 \leq i \leq n \\ \}$$

Für alle  $\delta_f^{(n)} = ([\delta_1], \dots, [\delta_n]) \in \Delta_f$  wird ein Endmakrozustand erzeugt. Seien dazu  $q_1, \dots, q_n$  die Zustände, die mit den Transitionen aus  $\delta_f^{(n)}$  erreichbar sind. Es wird ein Zustandstupel

$$\mathbf{q}_f = (q_1, \dots, q_m)$$

gebildet, dass allerdings diesem Zeitpunkt noch kein gültiger Endmakrozustand sein muss, da die mit einem Transitionstupel erreichten Zustände im Allgemeinen nicht paarweise verschieden sein müssen. Am Ende der Konstruktion wird daher eine Umbenennung vorgenommen. Das Tupel  $\mathbf{q}_f$  wird zu  $\mathcal{F}_2$  und zu  $\Omega_2$  hinzugefügt. Die Transitionen  $\delta_1, \dots, \delta_n$  werden zu  $\Delta_2$  hinzugefügt.

Zu jedem Transitionstupel  $\delta^{(n)} \notin \Delta_f$  wird ein Makrozustand wie folgt erzeugt:

Für die normalen Transitionen und  $\varepsilon$ -Transitionen  $\delta_1, \dots, \delta_m$  eines Transitionstupels  $\delta^{(n)} \in \Delta_1$  existieren Zustände  $q_1, \dots, q_m$  aus  $Q_1$  und ein  $j \in Var_1$  sodass:

$$\delta_i \subseteq (Reg(Q_1 \times Var_1) \times \Sigma \times \{q_i\} \times \{j\}) \cup (Q_1 \times Var \times \{\varepsilon\} \times \{q_i\} \times \{j\})$$

mit  $1 \leq i \leq m$ .  $\$$ -Transitionen sind nicht von Bedeutung. Für die  $\delta_1, \dots, \delta_m$  wird ein Zustandstupel

$$\mathbf{q} = (q_1, \dots, q_m)$$

gebildet. Auch hier muss  $\mathbf{q}$  zu diesem Zeitpunkt noch kein gültiger Makrozustand sein. Das Tupel  $\mathbf{q}$  wird zu  $\Omega_2$  hinzugefügt und die Transitionen  $\delta_1, \dots, \delta_m$  werden zu  $\Delta_2$  hinzugefügt. Die so entstandene Menge von Zustandstupeln wird anschließend in eine Menge von korrekten Makrozuständen umgewandelt. Dazu nutzen wir eine der Spracherweiterung *ext* nach Definition 5.1.1 ähnliche Umbenennungsfunktion  $f$ , die doppelt auftretende Zustände in einem Zustandstupel eindeutig benennt.

**Definition 5.2.4.** Sei  $(q_1, \dots, q_m)$  ein Tupel von Zuständen aus der Menge  $Q$ . Die Umbenennungsfunktion  $f : Q^m \rightarrow Q^m$  ordnet einem Zustand  $q_i$  in seinem  $j$ -ten Auftreten in  $(q_1, \dots, q_m)$  den Index  $j$  zu.

Die Umbenennungsfunktion wird auf alle Tupel angewandt, wir erhalten eine Menge  $\Omega'_2$  von gültigen Makrozuständen:

$$\Omega'_2 = \{\mathbf{q}' = (q_1, \dots, q_m) \mid \mathbf{q}' = f(\mathbf{q}), \mathbf{q} \in \Omega_2\}$$

Die Menge  $Q_2$  ist definiert durch  $Q_2 = \{q \mid q \in \mathbf{q}, \mathbf{q} \in \Omega'_2\}$ . Die Umbenennung wird ebenfalls auf die Menge  $\mathcal{F}_2$  angewendet, man erhält  $\mathcal{F}'_2$ .

War ein Zustand  $q$  vor der Umbenennung  $i$ -mal in einem Tupel enthalten, sind für diesen Zustand  $i$  Zustände  $q_1, \dots, q_i$  entstanden. Wir betrachten die Transitionen  $\delta = (L, a, (p, i)) \in \Delta_2$ , in denen  $q$  entweder erreicht wird oder in der Sprache der Transition definiert ist:

- $a \in \Sigma_2, i \in Var$
- Erster Fall: Für den regulären Ausdruck  $L \in Reg(Q \times Var)$  gilt:  $|L|_q \geq 1$
- Zweiter Fall:  $p = q$ .

In beiden Fällen werden  $i$  Kopien der Transition zu  $\Delta_2$  hinzugefügt, in denen  $q$  jeweils durch einen der Zustände  $q_1, \dots, q_i$  ersetzt wird. Damit steht grundsätzlich jede vorher gegebene Möglichkeit,  $q$  zu erreichen oder zu verlassen, zur Verfügung.

Mit  $\mathfrak{B}^{(n)} = (\Omega'_2, Q_2, Var, \Delta, \mathcal{F}')$  ist die Konstruktion des asynchronen unbeschränkt verzweigten Baumautomaten abgeschlossen.

Der resultierende Automat  $\mathfrak{B}$  kann jeden Lauf des Automaten  $\mathfrak{A}$  simulieren, da alle Transitionen eines Transitionstupels zur Verfügung stehen. Wenn mit  $\mathfrak{A}$  ein Transitionstupel  $(\delta_1, \dots, \delta_n)$  ausgeführt wird und mit den Transitionen die Zustände  $q_1, \dots, q_m$  erreicht werden, kann mit  $\mathfrak{B}$  genau ein Makrozustand  $(q_1, \dots, q_m)$  mit den - gegebenenfalls abgeänderten - Transitionen des Transitionstupels erreicht werden.  $\square$

Mit Lemma 5.2.2 und Lemma 5.2.1 folgt direkt Satz 5.2.1.  $\square$

### 5.2.2 Entscheidungsprobleme

Aus der Gleichmächtigkeit der transitions-synchronisierten Baumautomaten und der asynchronen Baumautomaten nach Satz 5.2.1 folgt direkt die Unentscheidbarkeit der folgenden Probleme.

**Korollar 5.2.1.** *Für zwei transitions-synchronisierte Baumautomaten  $\mathfrak{A}_1^{(n)}$  und  $\mathfrak{A}_2^{(n)}$  ist es im Allgemeinen unentscheidbar, ob*

- $\mathcal{R}(\mathfrak{A}_1^{(n)}) \cap \mathcal{R}(\mathfrak{A}_2^{(n)}) = \emptyset$

- $\mathcal{R}(\mathfrak{A}_1^{(n)}) \subseteq \mathcal{R}(\mathfrak{A}_2^{(n)})$
- $\mathcal{R}(\mathfrak{A}_1^{(n)}) = \mathcal{R}(\mathfrak{A}_2^{(n)})$
- $\mathcal{R}(\mathfrak{A}_1^{(n)}) = \mathcal{T}_\Sigma \times \dots \times \mathcal{T}_\Sigma$  .

Auch die Entscheidbarkeit des Nichtleerheitsproblems folgt unmittelbar aus der Entscheidbarkeit für asynchrone Baumautomaten. Wir geben hier dennoch einen Algorithmus an, der für einen gegebenen transitions-synchronisierten Baumautomaten das Problem entscheidet, da wir Abwandlungen dieses Algorithmus für Baumrelationen mit Anzahlbedingungen nutzen werden.

**Satz 5.2.2.** *Das Nichtleerheitsproblem für transitions-synchronisierte Baumautomaten ist entscheidbar.*

*Beweis.* Mit einem Transitionstupel kann eine akzeptierende Konfiguration eines transitions-synchronisierten Baumautomaten  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F)$  erreicht werden genau dann,

- wenn es keine  $\$$ -Transitionen enthält und
- mit seinen Transitionen nur akzeptierende Zustände erreichbar sind.

Es wird eine Menge  $\Delta_f$  bestehend aus allen Transitionstupeln, die den oben genannten Bedingungen genügen, berechnet:

$$\Delta_f = \{([\delta_{1_1}, \dots, \delta_{1_{r_1}}], \dots, [\delta_{n_1}, \dots, \delta_{n_{r_n}}]) \in \Delta \mid \begin{array}{l} \exists i \in Var : \forall j : \\ \delta_{j_k} \in (Reg(Q \times Var) \times \Sigma \times F \times Var) \\ \cup Q \times Var \times \{\varepsilon\} \times F \times Var \\ 1 \leq j \leq n, r_j \geq 1, 1 \leq j_k \leq r_j \end{array} \}$$

In dem folgenden Algorithmus bezeichnet die Menge  $T_h$  die Menge aller ausführbaren Transitionstupel zum Iterationszeitpunkt  $h$  und die Menge  $E_h$  die Menge der mit den Transitionstupeln erreichbaren Zustände zu diesem Zeitpunkt.

**IF**  $\Delta_f = \emptyset$  RETURN „leer“

$E_0 := \emptyset$

$T_0 := \emptyset$

$h := 0$

**REPEAT**

$h := h + 1$

**IF** „Transitionstupel  $\delta^{(n)} = ([\delta_{1_1}, \dots, \delta_{1_{r_1}}], \dots, [\delta_{n_1}, \dots, \delta_{n_{r_n}}]) \in \Delta$   
ist in Abhängigkeit von  $E_{h-1}$  ausführbar.“

**THEN**

$T_h := T_{h-1} \cup \{\delta^{(n)}\}$

**IF**  $T_h \cap \Delta_f \neq \emptyset$  RETURN „nicht leer“

**ELSE**  $E_h := E_{h-1} \cup \{(q, i) \in Q \times Var \mid$   
 $\exists \delta_{jk} \in \delta^{(n)} : \delta_{jk} \in Reg(Q \times Var) \times \Sigma \times \{q\} \times \{i\}, 1 \leq j \leq n, k \geq 1\}$

**UNTIL**  $E_h = E_{h-1}$

RETURN „leer“

Ein Transitionstupel  $\delta^{(n)}$  ist in Abhängigkeit von einer Menge  $E \subseteq Q \times Var$  ausführbar genau dann, wenn es für jeden in den Transitionen  $\delta_{1_1}, \dots, \delta_{1_{r_1}}, \dots, \delta_{n_1}, \dots, \delta_{n_{r_n}}$  definierten regulären Ausdruck ein  $w$  über der Menge  $E$  gibt, sodass  $w$  in der Sprache des Ausdruckes liegt, zudem müssen mit einem Transitionstupel vorher erreichte Zustände immer zusammen verwendet werden. Sei dazu  $q_1, \dots, q_m$  die Folge der Zustände, die mit einem Transitionstupel  $\delta_l^{(n)}$  erreicht werden, das in Iteration  $l$  der Menge  $T_l$  hinzugefügt wurde. Um auszudrücken, dass für jeden der Zustände  $q_1, \dots, q_m$  alle anderen genau einmal verwendet werden müssen, berechnen wir nach Satz 3.2.3 eine Presburger-Formel  $\varphi_l$ , die das Parikh-Bild der Sprache  $(q_1 \dots q_m)^*$  beschreibt.

In Iteration  $h$  wird überprüft, ob ein Transitionstupel

$$\delta_k^{(n)} = ([\delta_{1_1}, \dots, \delta_{1_{r_1}}], \dots, [\delta_{n_1}, \dots, \delta_{n_{r_n}}])$$

der Menge  $T_k$  hinzugefügt werden kann. Seien  $L_{1_1}, \dots, L_{n_{r_n}}$  alle in den Transitionen durch reguläre Ausdrücke definierte Sprachen.  $\varphi_L$  bezeichne eine Presburger-Formel, die das Parikh-Bild der Sprache  $L_{1_1} \dots L_{n_{r_n}}$ , der Konkatenation dieser Sprachen, beschreibt. Insgesamt muss in Iteration  $h$  die Formel

$$\varphi_L \wedge \varphi_1 \wedge \dots \wedge \varphi_{h-1}$$

erfüllbar sein, damit das Tupel  $\delta^{(n)}$  in Abhängigkeit der bisher erreichten Zustände ausführbar ist. Die Erfüllbarkeit der auftretenden Presburger-Formeln ist nach Satz 3.1.2 entscheidbar.

□

### 5.3 Transitions-Separierte Baumrelationen

Wie wir in Kapitel 5.1 festgestellt haben, entsprechen einstellige Relationen der Klasse  $\text{uRat}_1$  im Allgemeinen nicht den regulären Baumsprachen, während einstellige rationale

Wortrelationen genau reguläre Wortsprachen sind. Die Nicht-Regularität ist darin begründet, dass die beiden Automatenmodelle, durch die die unbeschränkt verzweigten Baumrelationen definiert sind, die Synchronisation innerhalb ein und desselben Baumes erlauben. Wir werden hier die Baumautomaten mit synchronisierten Transitionen nach Definition 5.2.2 so einschränken, dass dies nicht mehr möglich ist, und erhalten eine Teilklasse von  $\mathbf{uRat}$ , die *transitions-separierten Relationen über unbeschränkt verzweigten Bäumen*. Wir nennen diese Klasse  $\mathbf{SepuRat}$ .

**Definition 5.3.1.** Ein transitions-separierter Baumautomat auf  $n$ -Tupeln von unbeschränkt verzweigten Bäumen ist ein Tupel  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F)$  mit:

- $Q$  ist eine endliche Menge von Zuständen;
- $Var$  ist eine endliche Menge von Instanzvariablen;
- $\Sigma$  ist ein Beschriftungsalphabet;
- $F \subseteq Q$  ist eine Menge von Endzuständen;
- $\Delta$  ist eine Transitionsrelation mit

$$\Delta \subseteq \left( [ (Reg(Q \times Var) \times \Sigma \times Q \times Var) ] \cup \{\$\} \cup [ (Q \times Var \times \{\varepsilon\} \times Q \times Var) ] \right)^n .$$

Der Automat  $\mathfrak{A}$  führt einen Berechnungsschritt auf einem Tupel  $(t_1, \dots, t_n)$  zwischen zwei Konfigurationen  $c_1 : C_1 \rightarrow Q \times \mathbb{N}$  und  $c_2 : C_2 \rightarrow Q \times \mathbb{N}$  für zwei vollständige Schnitte  $C_1, C_2$  (Abbildung 5.5) durch, wenn die folgenden Voraussetzungen erfüllt sind.

- $C_1 = \{v_{11}, \dots, v_{kr_k}, v_{\varepsilon_1}, \dots, v_{\varepsilon_j}, v_{\$_1}, \dots, v_{\$_{j'}}\}$
- $C_2 = (C_1 \setminus \{v_{11}, \dots, v_{kr_k}\}) \cup \{v_1, \dots, v_k\}$
- Es existiert ein geeignetes Transitionstupel  $\delta^{(n)} = (\delta_1, \dots, \delta_n)$  mit
  - $k$  Transitionen  $(L_1, val(v_1), (q_1, o)), \dots, (L_k, val(v_k), (q_k, o))$  aus  $\delta^{(n)}$  mit  $L_i \in Reg(Q \times Var)$ ,  $1 \leq i \leq k$ ,  $o \in Var$ ,  $q_1, \dots, q_k \in Q$  und
  - $j$   $\varepsilon$ -Transitionen  $((q_{\varepsilon_1}, h_1), \varepsilon, (q_{\varepsilon_1'}, o)), \dots, ((q_{\varepsilon_j}, h_j), \varepsilon, (q_{\varepsilon_j'}, o))$  aus  $\delta^{(n)}$  und
  - $m$   $\$$ -Transitionen aus  $\delta^{(n)}$ , die keine Aktion ausführen,

und es gilt  $k, j, m \leq 0, k + j \geq 1, k + j + m = n$ . Für alle  $i \in (1, \dots, k)$  existieren Wörter  $w_i$  über  $Q \times EVar$  mit

$$w_i = (q_{i1}, o_{i1}) \dots (q_{ir_i}, o_{ir_i}) \in ext(L_i)$$

und es existieren geeignete Variablen und deren Instantiierung wie in Definition 5.2.2. Startkonfiguration, Lauf und Akzeptierbedingung sind ebenfalls wie für transitions-synchronisierte Automaten definiert.

Wir definieren die transitions-separierten Baumrelationen durch dieses Automatenmodell:

**Definition 5.3.2.** Eine Relation über unbeschränkt verzweigten Bäumen gehört zu der Klasse  $\text{SepuRat}$  genau dann, wenn sie von einem transitions-separierten Baumautomaten erkannt wird.

**Satz 5.3.1.** Die Klasse  $\text{SepuRat}_1$  entspricht der Klasse der regulären unbeschränkt verzweigten Baumsprachen.

*Beweis.* Die regulären Baumsprachen sind definiert durch unbeschränkt verzweigte Baumautomaten nach Definition 2.3.4 und die Klasse  $\text{SepuRat}_1$  durch transitions-separierte Baumautomaten auf 1-Tupeln von Bäumen. Es ist also zu zeigen, dass diese beiden Modelle gleich mächtig sind, wenn nur einstellige Relationen zugelassen sind.

**Lemma 5.3.1.** Für jeden unbeschränkt verzweigten Baumautomaten  $\mathfrak{A}$  gibt es einen transitions-separierten Baumautomaten  $\mathfrak{B}^{(1)}$ , sodass  $t \in \mathcal{T}(\mathfrak{A}) \Rightarrow (t) \in \mathcal{R}(\mathfrak{B}^{(1)})$ .

Sei  $\mathfrak{A} = (Q_1, \Sigma, \Delta_1, F_1)$  gegeben. Wir konstruieren einen transitions-separierten Baumautomaten  $\mathfrak{B}^{(1)} = (Q_2, \text{Var}_2, \Sigma, \Delta_2, F_2)$  mit

- $Q_2 = Q_1$ ,
- $\text{Var}_2 = \{i\}$ ,
- $F_2 = F_1$
- $([L_2, a, (q, i)]) \in \Delta_2 \Leftrightarrow (L_1, a, q) \in \Delta_1$  mit
  - $L_1 \in \text{Reg}(Q_1 \times \text{Var}_1), L_2 \in \text{Reg}(Q_2)$ ,
  - $L_2$  wird aus  $L_1$  gebildet, indem in dem regulären Ausdruck  $L_1$  für alle  $q \in Q_1$  der Zustand  $q$  durch  $(q, i)$  ersetzt wird.

Der Automat  $\mathfrak{B}^{(1)}$  enthält nur 1-Tupel von Transitionen, also wird auch nur eine Transition pro Berechnungsschritt ausgeführt. Für jede Transition  $(L_1, a, q) \in \text{Reg}(Q_1) \times \Sigma_1 \times Q_1$ , die für einen Lauf des Automaten  $\mathfrak{A}$  ausgeführt wird, kann Automat  $B$  ein Transitionstupel  $([L_2, a, (q, i)]) \in \text{Reg}(Q_2 \times \text{Var}_2) \times \Sigma_2 \times Q_2 \times \text{Var}$  ausführen. Die Synchronisation durch die Instanzvariablen hat keine Bedeutung, da keine mehr-elementigen Transitionstupel vorliegen.

**Lemma 5.3.2.** *Für jeden transitions-separierten Baumautomaten  $\mathfrak{B}^{(1)}$  gibt es einen unbeschränkt verzweigten Baumautomaten  $\mathfrak{A}$ , sodass  $(t) \in \mathcal{R}(\mathfrak{B}^{(1)}) \Rightarrow t \in \mathcal{T}(\mathfrak{A})$ .*

*Beweis.* Sei  $\mathfrak{B}^{(1)} = (Q_1, \text{Var}_1 \Sigma, \Delta_1, F_1)$  gegeben. Konstruiere  $\mathfrak{A} = (Q_2, \Sigma, \Delta_2, F_2)$  mit

- $Q_1 = Q_2$ ,
- $F_1 = F_2$
- $((L_2, a, q)) \in \Delta_2 \Leftrightarrow ((L_1, a, (q, i))) \in \Delta_1$  mit
  - $L_2 \in \text{Reg}(Q_2 \times \text{Var}_1), L_1 \in \text{Reg}(Q_1)$ ,
  - $L_2$  wird aus  $L_1$  gebildet, indem in dem regulären Ausdruck  $L_1$  für alle  $(q, i) \in Q_1 \times \text{Var}$  der Zustand  $(q, i)$  durch  $q$  ersetzt wird.

Da die Instanznummernvergabe für Automat  $\mathfrak{B}$  keine Bedeutung hat, kann jeder Lauf von  $\mathfrak{B}$  auch mit  $\mathfrak{A}$  durchgeführt werden. □

Da unbeschränkt verzweigte Baumautomaten und unbeschränkt verzweigte Baumautomaten mit synchronisierten Transitionen für 1-Tupel von Bäumen nach Lemma 5.3.1 und Lemma 5.3.2 gleich mächtig sind, folgt Satz 5.3.1. □

# Kapitel 6

## Baumrelationen mit Anzahlbedingungen

In diesem Kapitel werden die beiden zentralen Konzepte dieser Arbeit, *Anzahlbedingungen für Bäume* und *rationale Baumrelationen*, zusammengeführt. Wir definieren Relationen von Baumsprachen, sodass an die einzelnen Sprachen Anzahlbedingungen gestellt werden können, was zu neuen Klassen von Baumrelationen führt.

Die in Kapitel 5.2 vorgestellten transitions-synchronisierten Baumautomaten werden hier durchgängig als Definitionsgrundlage genutzt; wir kombinieren dieses Automatenmodell dann mit allen in Kapitel 3 vorgestellten Arten des Zählens auf Bäumen und erhalten sowohl *lokal* oder *global* zählende, als auch auf Zuständen der Automaten oder auf Symbolen des Alphabets zählende Automaten, die auf Tupeln von unbeschränkt verzweigten Bäumen arbeiten. Wir werden diese Modelle vergleichen und eine Hierarchie der entstanden Klassen von Baumrelationen mit Anzahlbedingungen angeben. Allgemein werden in diesem Kapitel *zählende* transitions-synchronisierte Baumautomaten definiert. Diese Automaten erkennen insgesamt die neue Klasse der *zählenden Baumrelationen*, die wir mit  $\text{Count} - \text{Rel}$  bezeichnen. Wir schreiben  $\text{Count} - \text{Rel}_n$ , wenn explizit  $n$ -stellige Relationen gemeint sind.

### 6.1 Lokale Anzahlbedingungen für Baumrelationen

Eine *Baumrelation mit lokalen Anzahlbedingungen* ist eine Relation von unbeschränkt verzweigten Baumsprachen, wobei an die einzelnen Sprachen lokale Anzahlbedingungen gemäß Kapitel 3 gestellt werden. Wir kombinieren dazu die transitions-synchronisierten Baumautomaten nach Definition 5.2.2 mit allen Automatenmodellen aus Kapitel 3.3.

#### 6.1.1 Presburger-Baumautomaten für Relationen

Die Presburger-Baumautomaten nach Definition 3.3.2 und Definition 3.3.3 arbeiten mit Presburger-regulären Ausdrücken über Zuständen des Automaten oder Symbolen des Alphabets. Da reguläre Ausdrücke in den Transitionen der transitions-synchronisierten

Automaten über  $Q \times Var$  definiert sind, benötigen wir eine abgeänderte Definition der Presburger-regulären Ausdrücke.

**Definition 6.1.1.** Ein Presburger-regulärer Ausdruck (*PReg*) über  $(Q \times Var)$  ist die Boolesche Kombination eines regulären Ausdrucks über  $(Q \times Var)$  und Presburger Formeln mit freien Variablen aus der Menge  $Y_Q = \{y_q \mid q \in Q\}$  oder mit freien Variablen aus der Menge  $Y_\Sigma = \{y_a \mid a \in \Sigma\}$

Nach Satz 3.3.1 sind Presburger-reguläre Ausdrücke entscheidbar. Durch diese abgeänderte Definition wird die Entscheidbarkeit nicht beeinflusst, da die Beweisschritte für diesen unter Vernachlässigung der Menge  $Var$  genauso durchgeführt werden können.

**Korollar 6.1.1.** *Es ist für einen Presburger-regulären Ausdruck  $\psi$  über  $Q \times Var$  entscheidbar, ob es ein Wort  $w$  gibt, sodass  $w \models \psi$ .*

Damit können wir die transitions-synchronisierten Presburger-Baumautomaten definieren.

**Definition 6.1.2.** Ein lokal auf Zuständen zählender transitions-synchronisierter Presburger-Baumautomat auf  $n$ -Tupeln von unbeschränkt verzweigten Bäumen ist ein Tupel  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F, \Phi_1, \dots, \Phi_n)$  mit:

- $Q$  ist eine endliche Menge von Zuständen;
- $Var$  ist eine endliche Menge von Instanzvariablen;
- $\Sigma$  ist ein Beschriftungsalphabet;
- $F \subseteq Q$  ist eine Menge von Endzuständen;
- $\Delta$  ist eine Transitionsrelation mit

$$\Delta \subseteq \left( \left[ \bigcup_{r \in \mathbb{N}^+} ((PReg(Q \times Var) \times \Sigma \times Q \times Var) \cup (Q \times Var \times \{\varepsilon\} \times Q \times Var))^r \right] \cup \{\$\} \right)^n.$$

- Die  $\Phi_1, \dots, \Phi_n$  sind Mengen von Presburger-Formeln mit freien Variablen nur aus  $Y_Q$ .

In dem  $i$ -ten Tupel von Transitionen eines Transitionstupels werden Presburger-Formeln aus der Menge  $\Phi_i$  mit  $1 \leq i \leq n$  verwendet. Dies gewährleistet eine eindeutige Zuordnung der Formelmengen zu Bäumen eines Eingabetupels. *Konfiguration*  $c$  und Instantiierungsfunktion  $I$  sind wir in Definition 5.2.2 definiert.

Der Automat  $\mathfrak{A}$  führt einen Berechnungsschritt auf einem Tupel  $(t_1, \dots, t_n)$  von unbeschränkt verzweigten Bäumen zwischen zwei Konfigurationen  $c : \mathcal{C}_1 \rightarrow Q \times Var$  und  $d : \mathcal{C}_2 \rightarrow Q \times Var$  durch, wenn es ein geeignetes Transitionstupel  $\delta^{(n)}$  und vollständige Schnitte  $C_1$  und  $C_2$  des Eingabetupels gibt, sodass:

- $\mathcal{C}_1 = \{v_{11}, \dots, v_{kr_k}, v_{\varepsilon_1}, \dots, v_{\varepsilon_j}, v_{\$1}, \dots, v_{\$j}, v_{n_1}, \dots, v_{n_{j'}}\}$
- $\mathcal{C}_2 = (C_1 \setminus \{v_{11}, \dots, v_{kr_k}\}) \cup \{v_1, \dots, v_k\}$
- In  $\delta^{(n)} = [(\delta_{1_1}, \dots, \delta_{1_{r_1}}), \dots, (\delta_{n_1}, \dots, \delta_{n_{r_n}})] \in \Delta$ ,  $r_i \geq 1$ ,  $1 \leq i \leq n$  gibt es
  - $k$  Transitionen  $(L_1, val(v_1), (q_1, o)), \dots, (L_k, val(v_k), (q_k, o))$  aus  $\delta^{(n)}$  mit  $L_i \in PReg(Q \times Var)$ ,  $1 \leq i \leq k$ ,  $o \in Var$ ,  $q_1, \dots, q_k \in Q$  und
  - $j$   $\varepsilon$ -Transitionen  $((q_{\varepsilon_1}, h_1), \varepsilon, (q_{\varepsilon'_1}, o)), \dots, ((q_{\varepsilon_j}, h_j), \varepsilon, (q_{\varepsilon'_j}, o))$  aus  $\delta^{(n)}$  und
  - $m' \leq m$   $\$$ -Transitionen aus  $\delta^{(n)}$ , die keine Aktion ausführen,

und es gilt  $k, j, m \geq 0$ ,  $k+j \geq 1$ . Damit wird mindestens ein neuer Zustand erreicht.

Die Konfiguration  $c$  bildet die Knoten  $v_{11}, \dots, v_{kr_k}$  auf Zustände  $q_{11}, \dots, q_{kr_k}$  ab. Diese Zustände bilden zusammen mit ihren zugehörigen Variablen für alle  $i \in (1, \dots, k)$  geeignete Wörter  $w_i$  über  $Q \times EVar$  mit

$$w_i = (q_{i1}, o_{i1}) \dots (q_{ir_i}, o_{ir_i}) \in ext(L_i)$$

und es existiert eine geeignete Instantiierung, sodass alle Knoten aus  $\mathcal{C}_1$ , die auf Zustände abgebildet werden, welche mit derselben Instanznummer kombiniert sind, entweder vollständig durch Transitionen verlassen werden oder nicht. Sei dazu für  $n \in \mathbb{N}$ :

$$A_n = \{v \in dom_{t_1, \dots, t_n} \mid c_n : v \mapsto (q, n), q \in Q \text{ beliebig}\}$$

und es gilt für alle  $n$ :

$$A_n \subseteq \{v_{11}, \dots, v_{kr_k}, v_{\varepsilon_1}, \dots, v_{\varepsilon_j}\} \Leftrightarrow A_n \cap \{v_{\$1}, \dots, v_{\$j}, v_{n_1}, \dots, v_{n_{j'}}\} = \emptyset.$$

*Startkonfiguration*, *akzeptierende Konfiguration* und *Lauf* sind wie in Definition 6.1.2 definiert.

Diese Automaten werden auch mit Anzahlbedingungen über den Symbolen des Eingabealphabetes definiert.

**Definition 6.1.3.** Ein lokal auf Symbolen zählender transitions-synchronisierter Presburger-Baumautomat auf  $n$ -Tupeln von unbeschränkt verzweigten Bäumen ist ein Tupel  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F, \Phi_1, \dots, \Phi_n)$  wie in Definition 6.1.2, wobei die Presburger-Formeln  $\Phi_1, \dots, \Phi_n$  nur freie Variablen aus der Menge  $Y_\Sigma$  mit  $Y_\Sigma = \{a \mid a \in \Sigma\}$  haben.

Mit diesen Automaten werden unter Vernachlässigung der Anzahlbedingungen genau die Relationen der Klasse `uRat` erkannt. Durch die Definition der Presburger-regulären Ausdrücke in den Transitionen werden Anzahlbedingungen an *einzelne* Bäume eines Eingabetupels gestellt. Die rationalen unbeschränkt verzweigten Baumrelationen werden dadurch erweitert; die hier vorgestellten Automaten definieren weitere Klassen von Baumrelationen, die wir im Vergleich der Modelle genau eingrenzen werden.

### 6.1.2 Parikh-Baumautomaten für Relationen

Um die rationalen Baumrelationen um das Konzept des *Parikh-Zählens* zu erweitern, benötigen wir zunächst abgeänderte Definitionen der Parikh-Abbildung und des Parikh-Wortautomaten, sodass Elemente aus  $Q \times Var$  ohne Beachtung der Menge  $Var$  abgebildet werden.

**Definition 6.1.4.** Seien  $\Sigma \times Var$  das kartesische Produkt eines endlichen Alphabets  $\Sigma$  und einer endlichen Variablenmenge  $Var$  und sei  $D$  eine nicht-leere Teilmenge von  $\mathbb{N}^n$  für ein  $n \in \mathbb{N}$ .

Die  $\Sigma \times Var$ -Projektion  $\Psi : (\Sigma \times Var \times D)^* \rightarrow (\Sigma \times Var)^*$  ist definiert durch

- $\Psi(a, i, \bar{d}) := (a, i)$ , für  $(a, i, \bar{d}) \in \Sigma \times Var \times D$
- $\Psi(uv) := \Psi(u)\Psi(v)$ , für  $u, v \in (\Sigma \times Var \times D)^*$ .

Die *erweiterte Parikh-Abbildung*  $\Phi : (\Sigma \times D)^* \rightarrow \mathbb{N}^n$  über  $\Sigma \times Var$  ist definiert durch

- $\Phi(a, i, \bar{d}) := \bar{d}$ , für  $(a, i, \bar{d}) \in \Sigma \times Var \times D$
- $\Phi(uv) := \Phi(u) + \Phi(v)$ , für  $u, v \in (\Sigma \times D)^*$ .

**Definition 6.1.5.** Ein Parikh-Automat über  $\Sigma \times Var$  der Dimension  $n \geq 1$  ist ein Paar  $(\mathfrak{A}, C)$  nach Definition 6.1.5, wobei der Automat  $\mathfrak{A}$  in einem Lauf die erweiterte Parikh-Abbildung über  $\Sigma \times Var$  und die  $\Sigma \times Var$ -Projektion aus Definition 6.1.4 nutzt.

Mit diesen Voraussetzungen definieren wir die Erweiterung des transitions-synchronisierten um Parikh-Automaten in den Transitionen.

**Definition 6.1.6.** Ein lokal über Zuständen zählender transitions-synchronisierter Parikh-Baumautomat auf  $n$ -Tupeln von unbeschränkt verzweigten Bäumen ist ein Tupel  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F, \Phi_1, \dots, \Phi_n)$  mit:

- $Q$  ist eine endliche Menge von Zuständen;
- $Var$  ist eine endliche Menge von Instanzvariablen;

- $\Sigma$  ist ein Beschriftungsalphabet;
- $F \subseteq Q$  ist eine Menge von Endzuständen;
- $\Delta$  ist eine Transitionsrelation mit

$$\Delta \subseteq \left( \left[ \bigcup_{r \in \mathbb{N}^+} \left( \left( \bigcup_{i=1}^n \Phi_i \times \Sigma \times Q \times Var \right) \cup (Q \times Var \times \{\varepsilon\} \times Q \times Var) \right)^r \right] \cup \{\$\} \right)^n .$$

- Die  $\Phi_1, \dots, \Phi_n$  sind Mengen von Parikh-Wortautomaten über  $Q \times Var$  für  $D \in \mathbb{N}^n, n \in \mathbb{N}$

In dem  $i$ -ten Tupel von Transitionen eines Transitionstupels werden Parikh-Automaten aus der Menge  $\Phi_i$  mit  $1 \leq i \leq n$  verwendet. Eine *Konfiguration*  $c$ , Instantiierungsfunktion  $I$ , *Startkonfiguration*, *akzeptierende Konfiguration* und *Lauf* sind wie in Definition 6.1.2 definiert.

Ein Berechnungsschritt auf einem Tupel  $(t_1, \dots, t_n)$  von unbeschränkt verzweigten Bäumen zwischen zwei Konfigurationen ist ebenfalls analog definiert, nur müssen die Kinder eines Knotens als Wort aufgefasst in der Sprache eines Parikh-Wortautomaten aus einer der Mengen  $\Phi_i, 1 \leq i \leq n$  sein.

Auch hier definieren wir Automaten, die auf Symbolen des Alphabets zählen.

**Definition 6.1.7.** Ein lokal auf Symbolen zählender transitions-synchronisierter Parikh-Baumautomat auf  $n$ -Tupeln von unbeschränkt verzweigten Bäumen ist ein Tupel  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F, \Phi_1, \dots, \Phi_n)$  wie in Definition 6.1.6, wobei die Parikh-Wortautomaten  $\Phi_1, \dots, \Phi_n$  über  $\Sigma \times Var \times D$  für  $D \in \mathbb{N}^n, n \in \mathbb{N}$  definiert sind.

### 6.1.3 Vergleich lokal zählender Baumrelationen

Wir vergleichen hier für Baumrelationen die Konzepte des Presburger-Zählens und des Parikh-Zählens. Offensichtlich ist, dass das Zählen auf Zuständen von Automaten nicht nur für Baumsprachen, sondern auch für Baumrelationen von größerer Aussagekraft ist als das Zählen auf Symbolen des Alphabets. Es sei zur Erklärung auf das Beispiel für Korollar 3.5.2 verwiesen.

**Korollar 6.1.2.** *Auf Symbolen zählende transitions-synchronisierte Baumautomaten sind im Allgemeinen von geringerer Ausdrucksstärke als solche, die auf Zuständen des Automaten zählen.*

**Lemma 6.1.1.** *Für jeden transitions-synchronisierten Presburger-Baumautomaten  $\mathfrak{A}^{(n)}$  gibt es einen transitions-synchronisierten Parikh-Baumautomaten  $\mathfrak{B}^{(n)}$  sodass*

$$(t_1, \dots, t_n) \in \mathcal{R}(\mathfrak{A}^{(n)}) \Leftrightarrow (t_1, \dots, t_n) \in \mathcal{R}(\mathfrak{B}^{(n)}) .$$

*Beweis.* Sei ein transitions-synchronisierter Presburger-Baumautomat  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F, \Phi_1, \dots, \Phi_n)$  gegeben. Konstruiere dazu einen transitions-synchronisierten Parikh-Baumautomaten  $\mathfrak{B}^{(n)} = (Q, Var, \Sigma, \Delta', F, \Phi'_1, \dots, \Phi'_n)$ . Die Zustandsmenge  $Q$ , die Variablenmenge  $Var$ , das Alphabet  $\Sigma$  und die Endzustandsmenge  $F$  werden übernommen. In jedem Transitionstupel

$$\delta^{(n)} = ([\delta_{1_1}, \dots, \delta_{1_{r_1}}], \dots, [\delta_{n_1}, \dots, \delta_{n_{r_n}}])$$

aus  $\Delta$  müssen die Presburger-regulären Ausdrücke durch Parikh-Automaten ersetzt werden. Dazu werden für alle  $i$  mit  $1 \leq i \leq n$  die Transitionen  $\delta_{i_j}$  des  $i$ -ten inneren Transitionstupels ausgewählt, für die gilt

$$\delta_{i_j} = (r, a, (q, o)) \in PReg(Q \times Var) \times \Sigma \times Q \times Var .$$

In jeder Transition wird der Presburger-reguläre Ausdruck  $r \in PReg(Q)$  nach Lemma 3.3.1 in disjunktive Normalform umgeformt. Es resultiert

$$r = r_1 \wedge \varphi_1 \vee \dots \vee r_n \wedge \varphi_m, \quad m \in \mathbb{N} .$$

Wir bilden für jeden regulären Ausdruck  $r_k$  mit  $1 \leq k \leq m$  einen nichtdeterministischen endlichen Automaten  $\mathcal{A}_k$  und bilden aus diesen Automaten den Vereinigungsautomaten  $\mathcal{A}_i = \bigcup_k \mathcal{A}_k$ . Für jede Formel  $\varphi_k$  berechnen wir nach Satz 3.1.3 eine semi-lineare Menge  $C_k$  und bilden die Menge  $C_i = \bigcup_k C_k$ . Wir ersetzen in der Transition  $(r, a, (q, o))$  den regulären Ausdruck  $r$  durch den Parikh-Wortautomaten  $(\mathcal{A}_i, C_i)$  und fügen  $(\mathcal{A}_i, C_i)$  der Menge  $\Phi_i$  hinzu. Nachdem diese Konstruktion für alle  $n$  inneren Transitionstupel durchgeführt wurde, ist die Menge  $\Delta'$  fertig gestellt. Sie enthält nur noch  $\varepsilon$ -Transition,  $\$$ -Transitionen und Transitionen mit Parikh-Wortautomaten. Diese Automaten sind in den zugehörigen Mengen  $\Phi_1, \dots, \Phi_n$  enthalten.  $\square$

Auch für zählende Baumrelationen gilt, dass für einen lokal zählenden Parikh-Automaten ein äquivalenter, lokal zählender Presburger-Automat konstruiert werden kann. Auf einen formalen Beweis wird hier verzichtet, es kann analog zu der oben stehenden Konstruktion der Beweis von Groz [16] unter Beachtung der Transitionstupel angepasst werden.

**Korollar 6.1.3.** *Für jeden transitions-synchronisierten Parikh-Baumautomaten  $\mathfrak{B}^{(n)}$  gibt es einen transitions-synchronisierten Presburger-Baumautomaten  $\mathfrak{A}^{(n)}$  sodass*

$$(t_1, \dots, t_n) \in \mathcal{R}(\mathfrak{A}^{(n)}) \Leftrightarrow (t_1, \dots, t_n) \in \mathcal{R}(\mathfrak{B}^{(n)}) .$$

Mit Lemma 6.1.1 und Korollar 6.1.3 formulieren wir den folgenden Satz.

**Satz 6.1.1.** *Transitions-synchronisierte Presburger-Baumautomaten und transitions-synchronisierte Parikh-Baumautomaten sind von gleicher Ausdrucksstärke.*

### 6.1.4 Entscheidungsprobleme

Wir betrachten die Entscheidungsprobleme für transitions-synchronisierte Presburger-Baumautomaten mit lokalen Anzahlbedingungen. Wir können alle Eigenschaften analog für Automaten mit Presburger-Bedingungen und Automaten mit Parikh-Automaten in den Transitionen schließen, da sowohl die über Symbolen zählenden als auch die über Zuständen zählenden Automaten jeweils gleich mächtig sind. Da alle Automaten eine Erweiterung der transitions-synchronisierten Baumautomaten sind, folgen nach Korollar 5.2.1 sofort die Unentscheidbarkeiten.

**Korollar 6.1.4.** *Für zwei transitions-synchronisierte Baumautomaten mit lokalen Anzahlbedingungen  $\mathfrak{A}_1^{(n)}$  und  $\mathfrak{A}_2^{(n)}$  ist es im Allgemeinen unentscheidbar, ob*

- $\mathcal{R}(\mathfrak{A}_1^{(n)}) \cap \mathcal{R}(\mathfrak{A}_2^{(n)}) = \emptyset$
- $\mathcal{R}(\mathfrak{A}_1^{(n)}) \subset \mathcal{R}(\mathfrak{A}_2^{(n)})$
- $\mathcal{R}(\mathfrak{A}_1^{(n)}) = \mathcal{R}(\mathfrak{A}_2^{(n)})$
- $\mathcal{R}(\mathfrak{A}_1^{(n)}) = \mathcal{T}_\Sigma \times \dots \times \mathcal{T}_\Sigma$  .

**Satz 6.1.2.** *Das Nichtleerheitsproblem für transitions-synchronisierte Baumautomaten mit lokalen Presburger Bedingungen ist entscheidbar.*

*Beweis.* Um das Nichtleerheitsproblem zu entscheiden, erweitern wir den Algorithmus für transitions-synchronisierte Baumautomaten aus Kapitel 5.2.2, der Mengen von ausführbaren Transitionstupeln berechnet. Wir geben den Algorithmus auch hier vollständig an.

Mit einem Transitionstupel kann eine akzeptierende Konfiguration eines transitions-synchronisierten Presburger-Baumautomaten  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F, \Phi_1, \dots, \Phi_n)$  erreicht werden genau dann,

- wenn es keine  $\$$ -Transitionen enthält und
- mit seinen Transitionen nur akzeptierende Zustände erreichbar sind.

Es wird eine Menge  $\Delta_f$  bestehend aus allen Transitionstupeln, die den oben genannten Bedingungen genügen, berechnet:

$$\Delta_f = \{([\delta_{1_1}, \dots, \delta_{1_{r_1}}], \dots, [\delta_{n_1}, \dots, \delta_{n_{r_n}}]) \in \Delta \mid \begin{array}{l} \exists i \in Var : \forall j : \\ \delta_{j_k} \in (PReg(Q \times Var) \times \Sigma \times F \times Var) \\ \cup Q \times Var \times \{\varepsilon\} \times F \times Var \\ 1 \leq j \leq n, r_j \geq 1, 1 \leq j_k \leq r_j \end{array} \}$$

Die Menge  $T_h$  bezeichnet die Menge aller ausführbaren Transitionstupel zum Iterationszeitpunkt  $h$  und  $E_h$  die Menge der mit den Transitionstupeln erreichbaren Zustände zu diesem Zeitpunkt.

**IF**  $\Delta_f = \emptyset$  RETURN „leer“

$E_0 := \emptyset$

$T_0 := \emptyset$

$h := 0$

**REPEAT**

$h := h + 1$

**IF** „Transitionstupel  $\delta^{(n)} = ([\delta_{1_1}, \dots, \delta_{1_{r_1}}], \dots, [\delta_{n_1}, \dots, \delta_{n_{r_n}}]) \in \Delta$  ist in Abhängigkeit von  $E_{h-1}$  ausführbar.“

**THEN**

$T_h := T_{h-1} \cup \{\delta^{(n)}\}$

**IF**  $T_h \cap \Delta_f \neq \emptyset$  RETURN „nicht leer“

**ELSE**  $E_h := E_{h-1} \cup \{(q, i) \in Q \times Var \mid$

$\exists \delta_{j_k} \in \delta^{(n)} : \delta_{j_k} \in PReg(Q \times Var) \times \Sigma \times \{q\} \times \{i\}, 1 \leq j \leq n, k \geq 1\}$

**UNTIL**  $E_h = E_{h-1}$

RETURN „leer“

Ein Transitionstupel  $\delta^{(n)}$  ist in Abhängigkeit von einer Menge  $E \subseteq Q \times Var$  ausführbar genau dann, wenn es für jeden in den Transitionen aus  $\delta^{(n)}$  definierten Presburger-regulären Ausdruck ein Wort  $w$  über der Menge  $E$  gibt, sodass  $w$  diesen Ausdruck erfüllt. Formal muss gelten:

$$\forall \delta_{j_k} = (r_{j_k}, a_{j_k}, (q, o)) : \exists w \in PReg(E) : w \models r_{j_k}, 1 \leq j \leq n, k \geq 1 .$$

Dies ist nach Korollar 6.1.1 entscheidbar.

Die vorher mit ein und demselben Transitionstupel erreichten Zustände müssen immer gemeinsam in gleicher Anzahl oder gar nicht ausgeführt werden. Sei dazu  $q_1, \dots, q_m$  die Folge der Zustände, die mit dem Transitionstupel  $\delta_l^{(n)}$  erreicht werden, das in Iteration  $l$  der Menge  $T_l$  hinzugefügt wurde. Um auszudrücken, dass für jeden der Zustände  $q_1, \dots, q_m$  alle anderen genau einmal verwendet werden müssen, berechnen wir nach Satz 3.2.3 eine Presburger-Formel  $\varphi_l$ , die das Parikh-Bild der Sprache  $(q_1 \dots q_m)^*$  beschreibt.

In Iteration  $h$  wird überprüft, ob ein Transitionstupel  $\delta_k^{(n)} = ([\delta_{1_{r_1}}, \dots, \delta_{1_{r_1}}], \dots, [\delta_{n_1}, \dots, \delta_{n_{r_n}}])$  der Menge  $T_k$  hinzugefügt werden kann. Seien  $r_{1_1}, \dots, r_{r_n}$  alle in den Transitionen definierten Presburger-regulären Ausdrücke.  $\varphi_r$  bezeichne eine Presburger-Formel, die das Parikh-Bild von  $r_{1_1} \cdot \dots \cdot r_{r_n}$ , der Konkatenation aller Presburger-regulären Ausdrücke, beschreibt. Insgesamt muss in Iteration  $h$  die Formel

$$\varphi_r \wedge \varphi_1 \wedge \dots \wedge \varphi_{h-1}$$

erfüllbar sein, damit das Tupel  $\delta^{(n)}$  in Abhängigkeit der bisher erreichten Zustände ausführbar ist. Die Erfüllbarkeit der auftretenden Presburger Formeln ist nach Satz 3.1.2 entscheidbar.  $\square$

## 6.2 Globale Anzahl-Bedingungen für Baumrelationen

Die lokal zählenden Automatenmodelle des vorangegangenen Kapitels werden nun um globale Anzahlbedingungen erweitert. Dabei werden wir die Konzepte von global zählenden Baumautomaten aus Kapitel 3.4 mit den transitions-synchronisierten Baumautomaten zusammenführen, die Entscheidbarkeitsprobleme betrachten, und sie hinsichtlich ihrer Mächtigkeit untersuchen und vergleichen.

### 6.2.1 Global zählende Presburger-Baumautomaten für Relationen

Die transitions-synchronisierten Presburger-Baumautomaten nach Definition 6.1.2 werden um globale Presburger-Formeln  $\phi_1, \dots, \phi_n$  erweitert. Die Formeln zählen über Symbolen, wenn für ein Eingabetupel  $t_1, \dots, t_n$  überprüft wird, ob  $t_i \models \phi_i$  für  $1 \leq i \leq n$ . Analog zählen sie über Zuständen, wenn für einen Lauf des Automaten  $(\rho_1, \dots, \rho_n)$  überprüft wird, ob  $\rho_i \models \phi_i$ . Die Variablenmengen sind wie in Kapitel 3.3 durch die Zustandsmenge bzw. das Eingabealphabet des Automaten gegeben:

$$Y_Q = \{y_q \mid q \in Q\} \text{ und } Y_\Sigma = \{y_a \mid a \in \Sigma\} .$$

**Definition 6.2.1.** Ein lokal und global auf Zuständen zählender transitions-synchronisierter Presburger-Baumautomat auf  $n$ -Tupeln von unbeschränkt verzweigten Bäumen ist ein Tupel  $(\mathfrak{A}^{(n)}, \phi_1, \dots, \phi_n)$ , wobei  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F, \Phi_1, \dots, \Phi_n)$  ein transitions-synchronisierter Presburger-Baumautomat nach Definition 6.1.2 ist. Die  $\phi_1, \dots, \phi_n$  sind Presburger-Formeln mit freien Variablen nur aus  $Y_Q$ . Ein Lauf des Automaten auf einem Eingabetupel  $(t_1, \dots, t_n)$  ist ein Tupel  $(\rho_1, \dots, \rho_n)$  von Bäumen. Ein Lauf ist *akzeptierend*, wenn er in einer akzeptierenden Konfiguration endet und für alle  $\rho_i$  gilt:

$$\rho_i \models \phi_i$$

für  $1 \leq i \leq n$  nach Definition 3.4.1.

**Definition 6.2.2.** Ein lokal und global auf Symbolen zählender transitions-synchronisierter Presburger-Baumautomat auf  $n$ -Tupeln von unbeschränkt verzweigten Bäumen ist ein Tupel  $(\mathfrak{A}^{(n)}, \phi_1, \dots, \phi_n)$ , wobei  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F, \Phi_1, \dots, \Phi_n)$  ein transitions-synchronisierter Presburger-Baumautomat nach Definition 6.1.2 ist. Die  $\phi_1, \dots, \phi_n$  sind Presburger-Formeln mit freien Variablen nur aus  $Y_\Sigma$ . Ein Lauf ist akzeptierend, wenn er in einer akzeptierenden Konfiguration endet, zusätzlich muss für jeden Baum  $t_i$  eines Eingabetupels  $(t_1, \dots, t_n)$  von unbeschränkt verzweigten Bäumen gelten, dass

$$t_i \models \phi_i$$

für  $1 \leq i \leq n$  nach Definition 3.4.1.

## 6.2.2 Global zählende Parikh-Baumautomaten für Relationen

Um lokal und global Parikh-zählende Baumsprachen zu definieren, fügen wir den transitions-synchronisierten Baumautomaten global auf Symbolen zählende Parikh-Baumautomaten hinzu, die entweder auf der Beschriftung der Bäume eines Eingabetupels oder auf der Beschriftung der Bäume des resultierenden Laufes zählen.

**Definition 6.2.3.** Ein lokal und global auf Zuständen zählender transitions-synchronisierter Parikh-Baumautomat auf  $n$ -Tupeln von unbeschränkt verzweigten Bäumen ist ein Tupel  $(\mathfrak{A}^{(n)}, (\mathfrak{A}_1, \mathcal{C}_1), \dots, (\mathfrak{A}_n, \mathcal{C}_n))$ , wobei  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F, \Phi_1, \dots, \Phi_n)$  ein transitions-synchronisierter Parikh-Baumautomat nach Definition 6.1.6 ist. Die  $(\mathfrak{A}_1, \mathcal{C}_1), \dots, (\mathfrak{A}_n, \mathcal{C}_n)$  sind global auf Symbolen zählende Parikh-Baumautomaten nach Definition 3.4.4. Der Automat akzeptiert ein Tupel  $t^{(n)} = (t_1, \dots, t_n)$ , wenn für jeden Baum  $\rho_i$  eines akzeptierenden Laufes  $(\rho_1, \dots, \rho_n)$  des transitions-synchronisierten Automaten gilt, dass

$$\rho_i \in \mathcal{T}((\mathfrak{A}_i, \mathcal{C}_i))$$

für  $1 \leq i \leq n$ .

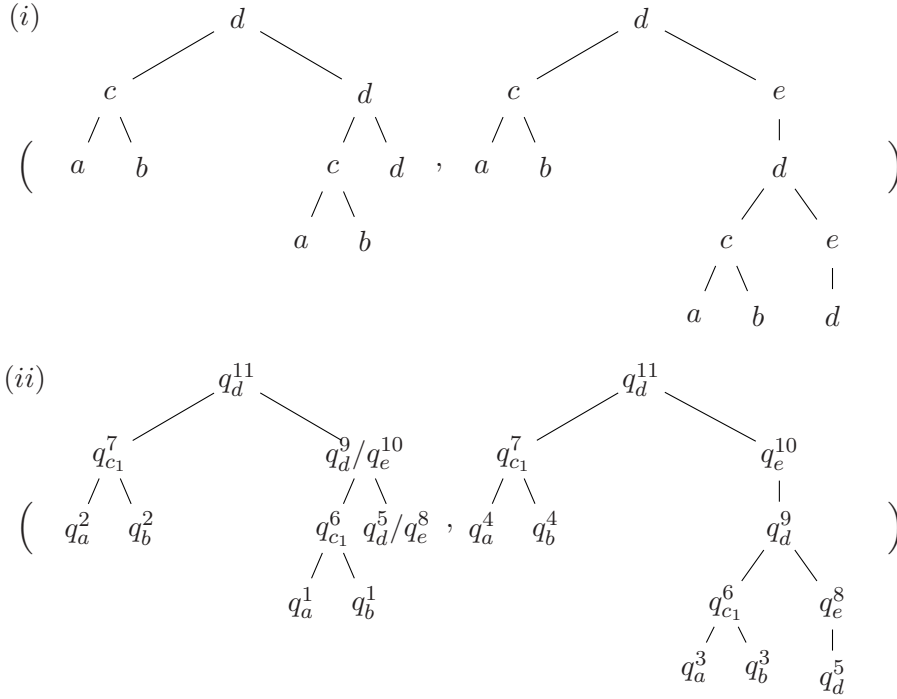


Abbildung 6.1: (i) Tupel  $(t_1, t_2) \in R_{\text{Count}}$ , (ii) Lauf  $(\rho_1, \rho_2)$  von  $\mathfrak{A}_{\text{Count}}^{(n)}$  auf  $(t_1, t_2)$

**Definition 6.2.4.** Ein lokal und global auf Symbolen zählender transitions-synchronisierter Parikh-Baumautomat auf  $n$ -Tupeln von unbeschränkt verzweigten Bäumen ist ein Tupel  $(\mathfrak{A}^{(n)}, (\mathfrak{A}_1, \mathcal{C}_1), \dots, (\mathfrak{A}_n, \mathcal{C}_n))$ , wobei  $\mathfrak{A}^{(n)} = (Q, \text{Var}, \Sigma \times D, \Delta, F, \Phi_1, \dots, \Phi_n)$  ein transitions-synchronisierter Parikh-Baumautomat nach Definition 6.1.6 ist. Die  $(\mathfrak{A}_1, \mathcal{C}_1), \dots, (\mathfrak{A}_n, \mathcal{C}_n)$  sind global auf Symbolen zählende Parikh-Baumautomaten nach Definition 3.4.4. Der Automat akzeptiert ein Tupel  $t^{(n)} = (t_1, \dots, t_n)$ , wenn für jeden Baum  $t_i$  gilt, dass

$$t_i \in \mathcal{T}((\mathfrak{A}, \mathcal{C}_i))$$

für  $1 \leq i \leq n$ .

Wir geben zusammenfassend ein Beispiel für einen lokal und global auf Zuständen zählenden transitions-synchronisierten Presburger-Baumautomaten an, sodass insbesondere das Zusammenwirken von Synchronisation zwischen Bäumen und Anzahlbedingungen klar wird.

**Beispiel 6.2.1.** Wir betrachten die unbeschränkt verzweigte Baumrelation  $R$  aus Beispiel 5.0.2. Diese Relation wird mit einer lokalen und einer globalen Anzahlbedingung versehen. Die zählende Baumrelation  $R_{\text{Count}}$  enthält alle Tupel  $(t_1, t_2) \in R$  sodass in  $t_1$  und  $t_2$  die folgenden Bedingungen erfüllt sind:

- Für die Kinder aller linken Knoten in  $t_1$  und  $t_2$ , die mit  $c$  beschriftet sind, gilt:

$$|a| = 1$$

- Für den gesamten Baum  $t_1$  gilt:

$$|c| = 2$$

Wir definieren diese Relation, indem wir den Automaten aus Beispiel 5.2.2 um entsprechende Presburger-reguläre Ausdrücke erweitern.

Sei  $\mathfrak{A}_{\text{Count}}^{(n)} = (Q, \text{Var}, \Sigma, \Delta, F, \Phi_1, \Phi_2, \phi_1, \phi_2)$  ein lokal und global zählenden transitions-synchronisierter Presburger-Baumautomat mit:

- $Q = \{q_a, q_b, q_c, q_{c_1}, q_d, q_e\}$
- $\text{Var} = \{i\}$
- $\Sigma = \{a, b, c, d, e\}$
- $F = \{q_d\}$
- $\Phi_1 = \{\varphi_a : y_{q_a} = 1\} = \Phi_2$
- $\phi_1 : y_{q_c} = 2, \phi_2 : \text{true}$
- $\Delta = \{ ( [ (a, (q_a, i)), (b, (q_b, i)) ] , \$ ), ( \$ , [ (a, (q_a, i)), (b, (q_b, i)) ] ), ( [ ((q_a, i)^*(q_b, i)^* \wedge \varphi_a, c, (q_{c_1}, i)) ] , [ ((q_a, i)^*(q_b, i)^*, c, (q_{c_1}, i)) ] ), ( [ (c, (q_c, i)) ] , [ (c, (q_c, i)) ] ), ( [ (d, (q_d, i)) ] , [ (d, (q_d, i)) ] ), ( [ ((q_d, i), \varepsilon, (q_e, i)) ] , [ ((q_d, i), e, (q_e, i)) ] ), ( [ ((q_{c_1}, i)(q_c, i)^*(q_e, i), d, (q_d, i)) ] , [ ((q_{c_1}, i)(q_c, i)^*(q_e, i), d, (q_d, i)) ] ) }$

In Abbildung 6.1 ist  $(i)$  ein Tupel  $(t_1, t_2) \in R_{\text{Count}}$  dargestellt sowie  $(ii)$  ein akzeptierender Lauf  $(\rho_1, \rho_2)$  von  $\mathfrak{A}_{\text{Count}}^{(n)}$  auf diesem Tupel. Es gilt  $\rho_1 \models \phi_1$ . Durch die innere Synchronisation der Symbole  $a$  und  $b$  fordert die lokale Anzahlbedingung implizit, dass auch  $|b| = 1$  gilt. Ebenso ist durch die Synchronisation der mit  $c$  beschrifteten Knoten zwischen den Bäumen  $t_1$  und  $t_2$  implizit für  $t_2$  gefordert, dass  $|c| = 2$ .

### 6.2.3 Vergleich global zählender Baumrelationen

Wir vergleichen die Ausdrucksstärke der global zählenden Baumrelationen. Zunächst folgt auch hier nach Korollar 3.5.2, dass das Zählen auf Zuständen mächtiger ist als das Zählen auf Symbolen.

**Korollar 6.2.1.** *Lokal und global auf Symbolen zählende transitions-synchronisierte Baumautomaten sind weniger ausdrucksstark als solche, die auf Zuständen des Automaten zählen.*

**Lemma 6.2.1.** *Für jeden lokal und global auf Symbolen zählenden transitions-synchronisierten Presburger-Baumautomaten  $\mathfrak{A}^{(n)}$  gibt es einen lokal und global zählenden transitions-synchronisierten Parikh-Baumautomaten  $\mathfrak{B}^{(n)}$ , sodass:*

$$(t_1, \dots, t_n) \in \mathcal{R}(\mathfrak{A}^{(n)}) \Leftrightarrow (t_1, \dots, t_n) \in \mathcal{R}(\mathfrak{B}^{(n)}) .$$

*Beweis.* Für einen gegebenen lokal und global auf Symbolen zählenden transitions-synchronisierten Presburger-Baumautomaten  $\mathfrak{A}^{(n)}$  konstruieren wir einen lokal und global zählenden transitions-synchronisierten Parikh-Baumautomaten  $\mathfrak{B}^{(n)}$ , indem wir die Konstruktion aus dem Beweis zu Lemma 6.1.1 anwenden. Um die globalen Presburger-Formeln  $\phi_1, \dots, \phi_n$  darzustellen, konstruieren wir für jede globale Presburger-Formel  $\phi_i$  einen global zählenden Parikh-Baumautomaten  $(\mathfrak{A}_{T_\Sigma}, \mathcal{C}_i)$ . Die Automatenkomponente  $\mathfrak{A}_{T_\Sigma}$  erkennt jede Baumsprache über dem Alphabet  $\Sigma$  und die semi-lineare Menge  $\mathcal{C}_i$  wird aus  $\phi_i$  nach Satz 3.1.3 berechnet. Da die Presburger-Formel nur eine Anzahlbedingung definiert, ist die Struktur eines Baumes für den Parikh-Automaten nicht von Bedeutung.  $\square$

Die nächste Eigenschaft folgt mit der Betrachtung einstelliger Relationen über unären Bäumen, da in diesem Fall weder die innere noch die äußere Synchronisation zwischen Tupелеlementen eine Rolle spielt. Das Gegenbeispiel des Beweises zu Lemma 3.5.4 ist dann auch für die lokal und global Parikh-zählenden Baumrelationen anwendbar, und es folgt:

**Korollar 6.2.2.** *Es gibt lokal und global auf Symbolen zählende Relationen von unbeschränkt verzweigten Baumsprachen, die Parikh-erkennbar, aber nicht Presburger-erkennbar sind.*

Es folgt direkt, dass bei Betrachtung global zählender Baumrelationen die Automaten, die mit Hilfe von Parikh-Automaten auf Symbolen zählen, mächtiger sind als die Automaten mit globalen Presburger-Formeln.

**Satz 6.2.1.** *Lokal und global auf Symbolen zählende transitions-synchronisierte Presburger-Baumautomaten sind von geringerer Ausdrucksstärke als lokal und global zählende transitions-synchronisierte Parikh-Baumautomaten.*

Es folgt insgesamt analog zu Korollar 3.5.1, dass transitions-synchronisierte Baumautomaten, die auf Zuständen zählen, mit globalen Presburger-Formeln oder global zählenden Parikh-Automaten die gleiche Ausdrucksstärke haben.

**Korollar 6.2.3.** *Lokal und global auf Zuständen zählende transitions-synchronisierte Presburger-Baumautomaten und lokal und global auf Zuständen zählende transitions-synchronisierte Parikh-Baumautomaten sind von der gleichen Ausdrucksstärke.*

## 6.2.4 Entscheidungsprobleme

Wir betrachten die Entscheidungsprobleme für die Klasse **Count** – **Rel** anhand der lokal und global auf Zuständen zählenden transitions-synchronisierten Presburger-Baumautomaten, da diese nach den Ergebnissen des vorherigen Abschnitts die mächtigste Klasse von Baumrelationen definieren.

Durch Vernachlässigung der Zählkomponente folgen direkt nach Korollar 5.2.1 die Unentscheidbarkeiten.

**Korollar 6.2.4.** *Für zwei transitions-synchronisierte Baumautomaten mit lokalen und globalen Anzahlbedingungen  $\mathfrak{A}_1^{(n)}$  und  $\mathfrak{A}_2^{(n)}$  ist es im Allgemeinen unentscheidbar, ob*

- $R(\mathfrak{A}_1^{(n)}) \cap R(\mathfrak{A}_2^{(n)}) = \emptyset$
- $R(\mathfrak{A}_1^{(n)}) \subset R(\mathfrak{A}_2^{(n)})$
- $R(\mathfrak{A}_1^{(n)}) = R(\mathfrak{A}_2^{(n)})$
- $R(\mathfrak{A}_1^{(n)}) = \mathcal{T}_\Sigma \times \dots \times \mathcal{T}_\Sigma$  .

Wir betrachten nun das Nichtleerheitsproblem für lokal und global zählende Presburger-Baumautomaten, und damit für die gesamte Klasse **Count** – **Rel**. Der für lokal zählende Automaten gegebene Algorithmus ist auch hier anwendbar, da die lokal zählende Komponente der Automaten übernommen wurde. Dies führt dazu, dass der Algorithmus für einen lokal und global zählenden Automaten entscheidet, ob es unter Vernachlässigung des globalen Zählens ein nicht leeres Tupel von Bäumen gibt, das in der Relation des Automaten liegt. Da die Automaten so definiert sind, dass die globalen Bedingungen im Anschluss an den Lauf des Automaten überprüft werden, überprüfen

wir die Erfüllbarkeit dieser Bedingungen im Anschluss an den Algorithmus. Intuitiv funktioniert das folgendermaßen:

Der Algorithmus fügt in jedem Iterationsschritt ein ausführbares Transitionstupel einer Menge von Transitionstupeln hinzu. Dazu wird für alle in dem Transitionstupel definierten Presburger-regulären Ausdrücke überprüft, ob diese mit den bisher erreichten Zuständen erfüllbar sind. Terminiert der Algorithmus mit dem Ergebnis, dass die lokal zählende Komponente des Automaten nicht leer ist, muss für alle betrachteten Presburger-regulären Ausdrücke in Abhängigkeit von den bis dahin erreichbaren Zuständen überprüft werden, ob bezüglich der Anzahlen von Symbolen bzw. Zuständen sowohl die Ausdrücke als auch die globalen Presburger-Formeln erfüllbar sind.

**Satz 6.2.2.** *Das Nichtleerheitsproblem für lokal und global auf Zuständen zählende Presburger-Baumautomaten ist entscheidbar.*

*Beweis.* Sei  $\mathfrak{A}^{(n)} = (Q, Var, \Sigma, \Delta, F, \Phi_1, \dots, \Phi_n, \phi_1, \dots, \phi_n)$  gegeben. Der folgende Algorithmus entscheidet, ob die Relation dieses Automaten leer ist. Zunächst wird analog zu Satz 6.1.2 die Menge der Transitionstupel berechnet, mit denen eine akzeptierende Konfiguration des Automaten erreicht werden kann.

$$\Delta_f = \{([\delta_{1_1}, \dots, \delta_{1_{r_1}}], \dots, [\delta_{n_1}, \dots, \delta_{n_{r_n}}]) \in \Delta \mid \begin{array}{l} \exists i \in Var : \forall j : \\ \delta_{j_k} \in (PReg(Q \times Var) \times \Sigma \times F \times Var) \\ \cup Q \times Var \times \{\varepsilon\} \times F \times Var \\ 1 \leq j \leq n, r_j \geq 1, 1 \leq j_k \leq r_j \end{array} \}$$

Die Menge  $T_h$  bezeichnet die Menge aller ausführbaren Transitionstupel zum Iterationszeitpunkt  $h$  und  $E_h$  die Menge der mit den Transitionstupeln erreichbaren Zustände zu diesem Zeitpunkt.

**IF**  $\Delta_f = \emptyset$  RETURN „leer“

$E_0 := \emptyset$

$T_0 := \emptyset$

$h := 0$

**REPEAT**

$h := h + 1$

**IF** „Transitionstupel  $\delta^{(n)} = ([\delta_{1_1}, \dots, \delta_{1_{r_1}}], \dots, [\delta_{n_1}, \dots, \delta_{n_{r_n}}]) \in \Delta$  ist in Abhängigkeit von  $E_{h-1}$  ausführbar.“

**THEN**

$T_h := T_{h-1} \cup \{\delta^{(n)}\}$

**IF**  $T_h \cap \Delta_f \neq \emptyset$  **AND**  
 „Die benutzten Presburger-regulären Ausdrücke  
 genügen den globalen Presburger-Formeln“  
**THEN RETURN** „nicht leer“  
**ELSE**  $E_h := E_{h-1} \cup \{(q, i) \in Q \times Var \mid$   
 $\exists \delta_{jk} \in \delta^{(n)} : \delta_{jk} \in PReg(Q \times Var) \times \Sigma \times \{q\} \times \{i\}, 1 \leq j \leq n, k \geq 1\}$   
**UNTIL**  $E_h = E_{h-1}$   
**RETURN** „leer“

Seien  $\psi_1, \dots, \psi_m$  mit  $m \in \mathbb{N}$  alle Presburger-regulären Ausdrücke aus Transitionen der Menge  $T_h$  zum Iterationszeitpunkt  $h$ . Alle diese Ausdrücke sind eindeutig den Tuppelementen zuzuordnen, auf die ihre Transitionen angewendet werden. Wir bilden daher für alle  $i$  mit  $1 \leq i \leq n$  Mengen

$$\begin{aligned}
 A_i := \{ \psi \mid & (\psi, a, (q, o)) \in T_h \subseteq PReg(Q \times Var) \times \Sigma \times Q \times Var \\
 & \text{und } (\psi, a, (q, o)) \text{ ist Element eines } i\text{-ten Tupels von Transitionen} \\
 & \text{aus einem Transitionstupel } \delta(n). \}
 \end{aligned}$$

Diese Mengen enthalten genau die Ausdrücke aus Transitionen, die auf die  $i$ -ten Elemente eines Eingabetupels angewendet werden. Nach Satz 3.2.3 wird eine Presburger-Formel  $\varphi_i$  berechnet, die das Parikh-Bild der Konkatenation aller Presburger-regulären Ausdrücke der Menge  $A_i$  beschreibt. Die globalen Presburger-Formeln sind erfüllbar, wenn für alle  $i$  gilt

$$\varphi_i \wedge \phi_i \text{ ist erfüllbar .}$$

□

### 6.3 Klassen von zählenden Baumrelationen

Wir nutzen abschließend die Ergebnisse über die Aussagestärke der in diesem Kapitel definierten Automatenmodelle zu einer Systematisierung der Klasse **Count – Rel** zählender Baumrelationen in verschiedene Teilklassen.

**Definition 6.3.1.** Die in Kapitel 6 vorgestellten Automaten auf Tupeln von unbeschränkt verzweigten Bäumen definieren die folgenden Teilklassen der Klasse **Count – Rel** zählender Baumrelationen:

- **LCount<sub>Alph</sub> – Rel**: lokal auf Symbolen des Alphabets zählende Baumrelationen
- **LCount<sub>States</sub> – Rel**: lokal auf Zuständen des Automaten zählende Baumrelationen

- $\text{PresGCount}_{\text{Alph}} - \text{Rel}$ : lokal und global auf Symbolen des Alphabets zählende Presburger-Baumrelationen
- $\text{ParikhGCount}_{\text{Alph}} - \text{Rel}$ : lokal und global auf Symbolen des Alphabets zählende Parikh-Baumrelationen
- $\text{GCount}_{\text{States}} - \text{Rel}$ : lokal und global auf Zuständen des Automaten zählende Baumrelationen

Die Vereinigung dieser Klassen definiert die Klasse  $\text{Count} - \text{Rel}$ .

Wir können eine nicht lineare Hierarchie dieser Klassen herstellen, da  $\text{LCount}_{\text{States}} - \text{Rel}$  nicht mit  $\text{PresGCount}_{\text{Alph}} - \text{Rel}$  und  $\text{GCount}_{\text{States}} - \text{Rel}$  vergleichbar ist.

**Korollar 6.3.1.** *Für die in Definition 6.3.1 definierten Klassen von zählenden Baumrelationen gilt:*

- $\text{LCount}_{\text{Alph}} - \text{Rel} < \text{LCount}_{\text{States}} - \text{Rel}$
- $\text{PresGCount}_{\text{Alph}} - \text{Rel} < \text{ParikhGCount}_{\text{Alph}} - \text{Rel} < \text{GCount}_{\text{States}} - \text{Rel}$
- $\text{LCount}_{\text{Alph}} - \text{Rel} < \text{PresGCount}_{\text{Alph}} - \text{Rel}$
- $\text{LCount}_{\text{States}} - \text{Rel} < \text{GCount}_{\text{States}} - \text{Rel}$
- $\text{GCount}_{\text{States}} - \text{Rel} = \text{Count} - \text{Rel}$



# Kapitel 7

## Zusammenfassung

In dieser Arbeit wurden zunächst verschiedene existierende Modelle von zählenden Baumautomaten aufgeführt und systematisiert. Die durch diese Automaten definierte Klasse **Count** von zählenden Baumsprachen wurde als Resultat der vergleichenden Analyse aller Modelle in eine Reihe von Teilklassen aufgegliedert.

Wir haben zwei äquivalente Ansätze für rationale Relationen auf beschränkt verzweigten Bäumen kennen gelernt, die natürliche Erweiterungen der rationalen Wortrelationen sind. Anschließend wurde ein existierendes Automatenmodell analysiert und aufgrund der Ergebnisse genutzt, um die Klasse **uRat** der rationalen Relationen über unbeschränkt verzweigten Bäumen definieren. In der Folge wurde ein neuer Automat entwickelt, der gerade die Relationen der Klasse **uRat** erkennt. Analog zu den beschränkt verzweigten Baumrelationen stellte sich heraus, dass die Klasse **uRat<sub>1</sub>** der einstelligen Baumrelationen nicht den regulären Baumrelationen entspricht. Diese Schwäche wurde durch eine Einschränkung des Automatenmodells mit der Definition der Klasse **SepuRat** der transitions-separierten Baumrelationen behoben. Es stellte sich heraus, dass alle entscheidbaren und unentscheidbaren Eigenschaften von unbeschränkt verzweigten Baumrelationen denen der rationalen Wortrelationen entsprechen.

Mit diesen beiden Konzepten der zählenden Baumsprachen und der rationalen Baumrelationen wurde die Klasse **Count – Rel** der zählenden Baumrelationen definiert. Dazu wurde die Definition der Klasse **uRat** mit den verschiedenen zählenden Baumautomaten kombiniert und man erhielt eine Vielzahl an zählenden Automaten, die bestimmte Relationen von unbeschränkt verzweigten Bäumen erkennen. Der Vergleich dieser Modelle führte zur Definition verschiedener Teilklassen von **Count – Rel**, deren Vereinigung diese Klasse definiert. Alle entscheidbaren Eigenschaften der rationalen Baumrelationen konnten auch hier gezeigt werden.

**Ausblick**

Sowohl die zählenden Baumsprachen als auch die rationalen Baumrelationen bieten noch viel Spielraum für weitere Forschung. Zunächst wäre eine Möglichkeit, für beide Konzepte die kontextfreien Baumsprachen zu betrachten. Es wäre interessant, was dies für eine Auswirkung auf Entscheidungsprobleme hat und welche Klassen resultieren. Zusätzlich wäre ein Vergleich zwischen den kontextfreien Baumsprachen und den einstelligen rationalen Baumrelationen anzustellen.

Die zählenden Baumrelationen sind in dieser Arbeit nur dahingehend behandelt worden, dass innerhalb ein und desselben Baumes gezählt werden kann. Es können aber durchaus Anzahlbedingungen an alle Bäume eines Eingabetupels gestellt werden, die Erweiterung der vorhandenen Automaten ist trivial. Interessant wäre dagegen, wie sich dann die Entscheidbarkeit insbesondere des Nichtleerheitsproblems verhält.

Für alle Automatenmodelle steht die Betrachtung einer deterministischen Variante aus. Radmacher hat ein deterministisches top-down Modell des asynchronen Automaten definiert [28]. Dadurch, dass die transitions-synchronisierten Baumautomaten eine feste Zuordnung von Transitionen zu Bäumen haben, ist durchaus auch ein deterministisches bottom-up Modell denkbar; dies gilt umso mehr für die transition-separierten Baumautomaten. Mit einem solchen Ansatz wäre dann eine Klasse deterministischer, unbeschränkt verzweigter Baumrelationen definierbar.

## Literaturverzeichnis

- [1] A. Arnold and M. Dauchet. Morphismes et bimorphismes d'arbres. *Theor. Comput. Sci.*, 20:33–93, 1982.
- [2] J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.
- [3] L. Berman. The complexity of logical theories. *Theoretical Computer Science*, 11(1):71–77, may 1980.
- [4] J. Berstel. *Transductions and Context-Free Languages*. Number 38 in Leitfäden der angewandten Mathematik und Mechanik. Teubner, Stuttgart, 1979.
- [5] A. Brüggemann-Klein, M. Murata, and D. Wood. Regular tree and regular hedge languages over unranked alphabets. Technical report hktust-tcsc-2001-05, HKUST Theoretical Computer Science Center Research, 2001.
- [6] C. Choffrut and U. Paris. Relations over words and logic: a chronology.
- [7] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: <http://www.grappa.univ-lille3.fr/tata>, 2007. release October, 12th 2007.
- [8] J. Cristau, C. Löding, and W. Thomas. Deterministic automata on unranked trees. In *FCT*, pages 68–79, 2005.
- [9] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Einführung in die mathematische Logik*. Wissenschaftliche Buchgesellschaft, Darmstadt, 2 edition, 1986.
- [10] S. Eilenberg. *Automata, Languages, and Machines*. Academic Press, Inc., Orlando, FL, USA, 1976.
- [11] C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9:47–68, 1965.

- [12] M. J. Fischer and M. O. Rabin. Super-exponential complexity of presburger arithmetic. pages 27–41, 1974.
- [13] P. C. Fischer and A. L. Rosenberg. Multitape one-way nonwriting automata. *J. Comput. Syst. Sci.*, 2(1):88–101, 1968.
- [14] F. Gécseg and M. Steinby. Tree automata. 1984.
- [15] S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas, and languages. *Pac. J. Math.*, 16:285–296, 1966.
- [16] B. Groz. Counting over Unranked Trees. Praktikumsbericht, RWTH Aachen, 2006.
- [17] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
- [18] W. Karianto. Parikh automata with pushdown stack. Diplomarbeit, RWTH Aachen, 2004.
- [19] F. Klaedtke. Bounds on the automata size for presburger arithmetic. *ACM Trans. Comput. Logic*, 9(2):1–34, 2008.
- [20] F. Klaedtke and H. Rueß. Parikh automata and monadic second-order logics with linear cardinality constraints. Technical Report 177, Institute of Computer Science at Freiburg University, 2002.
- [21] F. Neven. Automata, logic and xml. In *In CSL 02 - Annual Conference of the European Association for Computer Science Logic (invited talk)*, pages 2–26. Springer, 2002.
- [22] F. Neven. Automata theory for xml researchers. *Sigmod Record*, 31:2002, 2002.
- [23] M. Nivat. Transductions des langages de chomsky. *Ann. de l'Inst.*, 18:339–456, 1968.
- [24] R. Parikh. Language-generating devices. In *Quarterly Progress Report*, volume 60, pages 199–212. MIT, January 1960.
- [25] R. J. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966.
- [26] M. Presburger. Über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt. volume 4, pages 149–176, 1997.

- 
- [27] M. Rabin and D.Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):115–125, 1959.
- [28] F. G. Radmacher. Automatendefinierbare Relationen über Bäumen. Diplomarbeit, RWTH Aachen, 2007.
- [29] F. G. Radmacher. An automata theoretic approach to rational tree relations. In *SOFSEM*, pages 424–435, 2008.
- [30] J.-C. Raoult. A survey of tree transductions. In *Tree Automata and Languages*, pages 311–326. 1992.
- [31] J.-C. Raoult. Rational tree relations. *Bulletin of the Belgian Mathematical Society*, 4:149–176, 1997.
- [32] J. Sakarovitch. *Éléments de théorie des automates*. Vuibert, Paris, 2003.
- [33] H. Seidl, T. Schwentick, and A. Muscholl. Numerical document queries. In *22nd ACM Conference on Principles of Database Systems*, pages 155–166. ACM, 2003.
- [34] H. Seidl, T. Schwentick, and A. Muscholl. Counting in trees. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 575 – 612. Amsterdam University Press, 2008.
- [35] H. Seidl, T. Schwentick, A. Muscholl, and P. Habermehl. Counting in trees for free. In *ICALP*, pages 1136–1149, 2004.
- [36] Skolem. Über einige satzfunktionen in der arithmetik. In *Skrifter utgitt av Det Norske Videnskaps-Akademi i Oslo, I. Matematisk naturvidenskapelig klasse*, volume 7, pages 1–28, 1931.
- [37] J. Thatcher. A further generalization of finite automata. Technical report rc 1846, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1967.
- [38] J. Thatcher. There’s a lot more to finite automata theory than you would have thought. Technical report rc 2852 (13407), IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1970.
- [39] P. Wolper and B. Boigelot. On the construction of automata from linear arithmetic constraints. In *Proceedings of the 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science*, pages 1–19. Springer, 2000.