Institut für Formale Methoden der Informatik

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

# Verbotsmuster für Logik mit zwei Variablen über endlichen und unendlichen Wörtern

Amelie Heindl

| | |
|---|---|
| **Studiengang:** | Informatik |
| **Prüfer/in:** | PD Dr. Manfred Kufleitner |
| **Betreuer/in:** | PD Dr. Manfred Kufleitner |
| **Beginn am:** | 23. November 2022 |
| **Beendet am:** | 23. Mai 2023 |

## Kurzfassung

Die Varietät **DA**, die mit der Logik erster Stufe über zwei Variablen übereinstimmt, bildet das höchste Level der Trotter-Weil-Hierarchie. Varietäten in dieser Hierachie können algebraisch über $\omega$-Gleichungen charakterisiert werden und haben teilweise äquivalente Klassen in der Quantorenalternierungshierarchie innerhalb $FO^2$. Um Sprachen mit diesen Klassen in Verbindung zu bringen, ist es möglich, das syntaktische Monoid auszurechnen und für alle möglichen Belegungen die $\omega$-Gleichung zu überprüfen. Aus komplexitätstheoretischer Sicht ist dieser Ansatz aufwändig und diese Arbeit befasst sich mit Kriterien, mit denen man die gesuchte Verbindung direkt auf den Automaten der Sprachen prüfen kann. Das zentrale Konzept stellen dabei Verbotsmuster dar. In dieser Arbeit wird eine Definition für Verbotsmuster bei nicht notwendigerweise minimalen DEAs und CMAs gegeben. Damit werden explizite Verbotsmuster für Sprachen über endlichen Wörtern für die Varietäten an den Enden der Hierarchie konstruiert und ein allgemeines Schema für die inneren Level angepasst. Aus diesen Mustern werden Verbotsmuster für fin-syntaktische Monoide von Sprachen über unendlichen Wörtern abgeleitet. Zusammen mit einem Verbotsmuster, das die Zugehörigkeit zu **DA** für inf-syntaktische Monoide entscheidet, lässt sich damit die Trotter-Weil-Hierarchie vollständig für DEAs und CMAs charakterisieren. Für alle konstruierten Verbotsmuster werden NL- und P-Algorithmen erstellt, die Automaten auf das Enthalten dieser Verbotsmuster prüfen. Dies beinhaltet sowohl Algorithmen, bei denen ein festes Verbotsmuster gegeben ist, als auch Algorithmen, bei denen es Teil der Eingabe ist. Für die Varietäten der Trotter-Weil-Hierarchie existieren auch Verbotsmuster mit Teilwortbeziehungen, die von Henriksson und Kufleitner definiert wurden. Die beiden Verbotsmusterarten werden in dieser Arbeit verglichen. Dabei werden insbesondere die Größen der Muster, die Komplexitätsschranken der Algorithmen und die gefundenen Zeugen berechnet. Es ergibt sich ein Größenvorteil der Teilwortmuster gegenüber den Faktormustern bei vergleichbaren Komplexitäten der Algorithmen.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Verzeichnis der Algorithmen

# Abkürzungsverzeichnis

**CMA** Carton-Michel Automat. 22

**DEA** deterministischer endlicher Automat. 15

**EXP** deterministisch exponentielle Zeit. 63

**FO** Logik erster Stufe. 15

**$FO^2$** Logik erster Stufe über zwei Variablen. 15

**MSO** monadische Logik zweiter Stufe. 15

**NEA** nichtdeterministischer endlicher Automat. 22

**NL** nichtdeterministisch logarithmischer Platz. 45

**P** deterministisch polynomielle Zeit. 52

# 1 Einleitung

Formale Sprachen stellen einen wichtigen Teil der theoretischen Informatik dar und bilden die Grundlage für eine Vielzahl von Anwendungen, beispielsweise in der Programmierung. Ein besonders interessantes Forschungsgebiet ist hierbei die eingeschränkteste Klasse der Chomsky Hierarchie, die Klasse der regulären Sprachen. Gängige Methoden zur Darstellung regulärer Sprachen sind deterministische endliche Automaten (DEAs), Grammatiken und reguläre Ausdrücke.

Die Menge der regulären Sprachen kann aber auch über algebraische beziehungsweise logische Kriterien definiert werden. Einerseits zeigten Myhill und Nerode mithilfe der Nerode Kongruenz, dass die regulären Sprachen mit den endlichen Monoiden korrespondieren [15], andererseits bewiesen Büchi, Elgot und Trakhtenbrot, dass sie mit den in monadischer Logik zweiter Stufe (MSO) definierbaren Formeln übereinstimmen [2], [7], [24]. Es wurde also eine Äquivalenz zwischen einer Sprachklasse, einer Monoidvarietät und einem Logikfragment gefunden. Durch das Verbinden dieser Konzepte aus den Bereichen der formalen Sprachen, der Algebra und der Logik lassen sich vielseitige Ergebnisse erzielen und es wurde nach weiteren derartigen Zusammenhängen bei eingeschränkteren Klassen geforscht. Resultate von Schützenberger, sowie von McNaughton und Papert führten zum Entdecken dieser Beziehung zwischen sternfreien Sprachen, aperiodischen Monoiden und Logik erster Stufe (FO) [19], [14]. Aus bestimmten Untervarietäten der Varietät der aperiodischen Monoide, bildeten Trotter und Weil eine Hierarchie [25]. Für einige Level dieser Hierarchie konnte eine Verbindung zu Logikfragmenten innerhalb der Logik erster Stufe über zwei Variablen ($FO^2$) gefunden werden, was sie zu einem interessanten Forschungsthema macht, mit dem auch diese Arbeit sich befasst.

Ein zentrales Konzept, das dabei verwendet wird, sind Verbotsmuster. Das sind Graphen, die in Automaten abgebildet werden können und durch eine erfolgreiche Abbildung auf die Abwesenheit gewisser Eigenschaften der vom Automaten erkannten Sprache schließen lassen. Mithilfe dieser Verbotsmuster werden in dieser Arbeit Sprachen mit den Varietäten aus der Trotter-Weil-Hierarchie zusammengebracht. Dabei werden sowohl Sprachen mit endlichen als auch mit unendlichen Wörtern untersucht. Vergleichend zu den dabei erreichten Resultaten, werden auch eingeschränktere Verbotsmuster betrachtet, welche Henriksson und Kufleitner für Varietäten der Trotter-Weil-Hierarchie definieren [10], [9].

Die Arbeit ist folgendermaßen gegliedert. In Kapitel 2 werden die benötigten Vorkenntnisse und Definitionen erklärt und in Kapitel 3 werden Verbotsmuster als Mittel zur Charakterisierung von Sprachen vorgestellt. Kapitel 4 definiert Verbotsmuster für die Level der Trotter-Weil-Hierarchie über endlichen und unendlichen Wörtern. Um diese Verbotsmuster in Automaten zu finden, werden in Kapitel 5 Algorithmen angegeben, welche das Vorhandensein eines Musters in logarithmisch beschränktem Platz entscheiden. Kapitel 6 behandelt den Vergleich zwischen den in Kapitel 4 definierten Mustern und den Verbotsmustern mit Teilwortbeziehungen von Henriksson und Kufleitner. In Kapitel 7 wird die Frage betrachtet, wie es sich auf die Algorithmen auswirkt, wenn

sie Verbotsmuster als Eingabe bekommen, statt ein festes Verbotsmuster zugeordnet zu haben. Abschließend werden die Ergebnisse in Kapitel 8 zusammengefasst und in den Forschungskontext eingeordnet.

# 2 Grundlagen

In diesem Kapitel werden für diese Arbeit notwendige Definitionen und Zusammenhänge aus den Bereichen Algebra, Logik und formale Sprachen präsentiert.

## 2.1 Halbgruppen und Monoide

**Definition 2.1 (Halbgruppe)** *Eine Halbgruppe ist ein Tupel* $(S, \cdot)$ *bestehend aus einer Menge* $S$ *und einer assoziativen zweistelligen Verknüpfung* $\cdot : S \times S \to S$.

**Definition 2.2 (Monoid)** *Ein Monoid ist ein Tupel* $(M, \cdot, 1)$ *bestehend aus einer Menge* $M$*, einer assoziativen zweistelligen Verknüpfung* $\cdot : M \times M \to M$ *und einem neutralen Element 1 mit der Eigenschaft* $\forall m \in M : 1 \cdot m = m = m \cdot 1$.

Vereinfacht werden Halbgruppen und Monoide mit dem Symbol Ihrer Menge bezeichnet.

Das Monoid, das aus einer Halbgruppe $S$ durch Zufügen eines neutralen Elements entsteht, wird mit $S^1$ bezeichnet.

In jeder endlichen Halbgruppe $S$ gibt es einen kleinsten Exponenten, für den jedes Element zu einem Idempotenten wird. Dieser wird mit $\omega$ oder $\omega_S$ bezeichnet. Es gilt also $\forall s \in S : s^\omega s^\omega = s^\omega$.

Zur Charakterisierung von Halbgruppen werden Greens Relationen genutzt. Sei $S$ eine Halbgruppe und $u, v \in S$. Greens Relationen gruppieren die Elemente von $S$ und geben Aufschluss über die Teilbarkeit. Es handelt sich um die folgenden Relationen:

$$
\begin{aligned}
u \mathcal{J} v &\iff S^1 u S^1 = S^1 v S^1 \\
u \mathcal{R} v &\iff u S^1 = v S^1 \\
u \mathcal{L} v &\iff S^1 u = S^1 v \\
u \mathcal{H} v &\iff u \mathcal{R} v \wedge u \mathcal{L} v \\
u \mathcal{D} v &\iff \exists w \in S : u \mathcal{L} w \wedge w \mathcal{R} v
\end{aligned}
$$

Es gibt auch die Relationen $\leq_\mathcal{J}, \leq_\mathcal{R}$ und $\leq_\mathcal{L}$, wobei nur Teilmengen- statt Gleichheitsbeziehungen gefordert sind. Eine Halbgruppe $S$ ist $\mathcal{C}$-trivial, falls die $\mathcal{C}$-Relation die Identität auf $S$ ist für $\mathcal{C} \in \{\mathcal{R}, \mathcal{L}, \mathcal{J}, \mathcal{H}, \mathcal{D}\}$.

Egg-Box-Diagramme visualisieren die Äquivalenzklassen, die von Greens Relationen erzeugt werden. Das Egg-Box Diagramm für eine Halbgruppe besteht aus Rechtecken für jede $\mathcal{D}$-Klasse (bei endlichen Halbgruppen sind diese identisch zu den $\mathcal{J}$-Klassen), die übereinander geordnet sind entsprechend $\leq_\mathcal{D}$. In jedem Rechteck sind die $\mathcal{R}$-Klassen als Zeilen, die $\mathcal{L}$-Klassen als Spalten

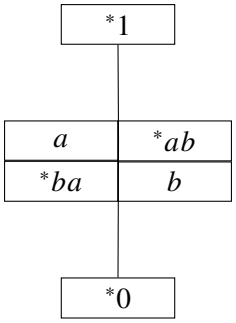



**Abbildung 2.1:** Egg-Box-Diagramm für $Synt((ab)^*) = \{1, a, b, ab, ba, 0\}$.

und die $\mathcal{H}$-Klassen als resultierende Zellen. Jede $\mathcal{H}$-Klasse, die ein Idempotentes enthält, ist mit $*$ markiert. Abbildung 2.1 zeigt beispielhaft das Egg-Box-Diagramm für das syntaktische Monoid von $(ab)^*$ [5].

## 2.2 Logik

Sei $X$ eine Variablenmenge und $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ ein endliches Alphabet. In dieser Arbeit betrachten wir Logik über der Signatur $\{\leq, \sigma_1, \ldots, \sigma_n\}$. Das Fragment FO über dieser Signatur enthält die atomaren Formeln $\top$, $\bot$, $(x = y)$, $(x \leq y)$ und $\lambda(x) = \sigma$ mit $x, y \in X$, $\sigma \in \Sigma$. Außerdem enthält FO, falls $\phi, \psi$ in FO sind, auch die Formeln $\neg\phi$, $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $\exists\phi$ und $\forall\phi$. $\mathrm{FO}^2$ über dieser Signatur hat die zusätzliche Einschränkung $|X| = 2$, was bedeutet, dass nur zwei Variablen in jeder Formel verwendet werden dürfen.

Formeln werden über Wörtern aus $\Sigma^*$ ausgewertet und die Variablen entsprechen Positionen innerhalb der Wörter. $\lambda(x) = \sigma$ bedeutet dann, dass an Position $x$ der Buchstabe $\sigma$ steht und $x < y$ bedeutet, dass Position $x$ links von Position $y$ ist. Ein Wort $w \in \Sigma^*$ erfüllt eine Formel $\phi$, wenn man die Variablen in $\phi$ mit Positionen in $w$ belegen kann, sodass $\phi$ sich zu $\top$ auswertet. Eine Formel ohne freie Variablen wird Satz genannt und kann auch von Wörtern erfüllt werden. Jeder Satz $\phi$ definiert somit eine Sprache $L = \{w \in \Sigma^* \mid w \text{ erfüllt } \phi\}$.

Mit De Morgans Gesetz kann jede $\mathrm{FO}^2$-Formel in eine äquivalente Formel umgeformt werden, bei der Negationen nur noch auf atomare Formeln angewandt werden. Bei diesen Formeln lässt sich die Zahl der Alternierungen zwischen Existenz- und Allquantoren bestimmen. Mit $\mathrm{FO}^2_m$ wird die Menge der $\mathrm{FO}^2$-Formeln bezeichnet, die höchstens $m$ Quantorenalternierungen haben. Diese Fragmente $\mathrm{FO}^2_0 \subseteq \mathrm{FO}^2_1 \subseteq \mathrm{FO}^2_2 \subseteq \ldots$ bilden eine Hierarchie innerhalb von $\mathrm{FO}^2$, die Quantorenalternierungshierarchie genannt wird.

## 2.3 Varietäten und die Trotter-Weil-Hierarchie

**Definition 2.3 (Varietät)** *Eine Varietät* **V** *ist eine Menge von endlichen Monoiden bzw. Halbgruppen, die abgeschlossen ist unter Division und endlichen direkten Produkten.*

Zu jeder Monoidvarietät gibt es auch eine korrespondierende Sprachenvarietät, die jene Sprachen enthält, deren syntaktisches Monoid in der Varietät ist [6].

**Definition 2.4 ($\omega$-Term)** *Sei $X$ eine Variablenmenge. Dann ist jedes Element $x \in X$, sowie das leere Wort $\varepsilon$, ein $\omega$-Term. Für $\omega$-Terme $u$ und $v$, sind $uv$ und $u^\omega$ auch $\omega$-Terme.*

Mit $\omega$-Termen können Varietäten charakterisiert werden. Hierfür muss definiert werden, was es für ein Monoid bedeutet, eine Gleichung von $\omega$-Termen zu erfüllen. Seien $u$ und $v$ $\omega$-Terme über der Variablenmenge $X$ und sei $M$ ein endliches Monoid. Dann erfüllt $M$ die Gleichung $u = v$, falls $\varphi(u) = \varphi(v)$ für jeden Homomorphismus $\varphi : X^* \to M$ gilt. $\varphi$ lässt sich hierbei durch $\varphi(u^\omega) = \varphi(u)^{\omega_M}$ auf die $\omega$-Terme ausweiten mit $\omega_M$ als dem Idempotenz-Exponenten für $M$. Nun bezeichnet man mit $[\![u = v]\!]$ die Varietät, welche aus allen Monoiden besteht, die die Gleichung $u = v$ erfüllen.

Bevor die benötigten Varietäten vorgestellt werden können, sind noch weitere Definitionen notwendig.

**Definition 2.5 (aperiodisch)** *Eine endliche Halbgruppe $S$ ist aperiodisch, falls $\forall x \in S : x^\omega = x^{\omega+1}$ für ein $\omega \in \mathbb{N}$ gilt.*

**Definition 2.6 (idempotent)** *Eine Halbgruppe $S$ ist idempotent, falls jedes Element $s \in S$ idempotent ist, also $\forall s \in S : s^2 = s$ gilt. Die Menge aller idempotenten Elemente aus $S$ wird mit $E(S)$ bezeichnet.*

**Definition 2.7 (reguläre $\mathcal{D}$-Klasse)** *Eine $\mathcal{D}$-Klasse ist regulär, falls sie ein reguläres Element enthält. Das bedeutet, es existieren Elemente $x, y$ in dieser $\mathcal{D}$-Klasse mit $xyx = x$.*

**Definition 2.8 (orthodox)** *Eine Halbgruppe $S$ heißt orthodox, falls $E(S)$ eine Unterhalbgruppe von $S$ bildet.*

Nun werden in Tabelle 2.1 einige für diese Arbeit benötigte Varietäten endlicher Monoide definiert.

| Symbol | Bezeichnung | Zugehörige $\omega$-Gleichungen |
|---|---|---|
| **A** | aperiodische Monoide (entspricht auch den $\mathcal{H}$-trivialen Monoiden) | $[\![x^\omega x = x^\omega]\!]$ |
| **DS** | Monoide, bei denen jede reguläre $\mathcal{D}$-Klasse eine Halbgruppe ist | $[\![((xy)^\omega x(xy)^\omega)^\omega = (xy)^\omega]\!]$<br>$[\![((xy)^\omega y(xy)^\omega)^\omega = (xy)^\omega]\!]$<br>$[\![((xyz)^\omega y(xyz)^\omega)^\omega = (xyz)^\omega]\!]$ |
| **DO** | Monoide, bei denen jede reguläre $\mathcal{D}$-Klasse eine orthodoxe Halbgruppe ist | $[\![(xy)^\omega (yx)^\omega (xy)^\omega = (xy)^\omega]\!]$ |
| **DA** | Monoide, bei denen jede reguläre $\mathcal{D}$-Klasse eine aperiodische Halbgruppe ist | $[\![(xy)^\omega x(xy)^\omega = (xy)^\omega]\!]$<br>$[\![(xy)^\omega y(xy)^\omega = (xy)^\omega]\!]$<br>$[\![(xyz)^\omega y(xyz)^\omega = (xyz)^\omega]\!]$ |
| **G** | endliche Gruppen | $[\![x^\omega = 1]\!]$ |
| $\mathbf{J_1}$ | idempotente und kommutative Monoide | $[\![x^2 = x, xy = yx]\!]$ |
| **R** | $\mathcal{R}$-triviale Monoide | $[\![(xy)^\omega x = (xy)^\omega]\!]$<br>$[\![(xyz)^\omega y = (xyz)^\omega]\!]$ |
| **L** | $\mathcal{L}$-triviale Monoide | $[\![y(xy)^\omega = (xy)^\omega]\!]$<br>$[\![y(xyz)^\omega = (xyz)^\omega]\!]$ |
| **J** | $\mathcal{J}$-triviale Monoide | $[\![(xy)^\omega x(ts)^\omega = (xy)^\omega s(ts)^\omega]\!]$ |
| **D** | definite Halbgruppen | $[\![yx^\omega = x^\omega]\!]$ |
| **K** | reverse definite Halbgruppen | $[\![x^\omega y = x^\omega]\!]$ |
| **B** | idempotente Halbgruppen | |

**Tabelle 2.1:** Spezifizierung einiger relevanter Varietäten.

Insbesondere die Varietät **DA** stellt ein interessantes Forschungsgebiet dar. Thérien und Wilke zeigten, dass diese Varietät mit der $\mathrm{FO}^2$-Logik übereinstimmt [23]. Innerhalb von **DA** lässt sich eine Hierarchie aufbauen, welche aus den eindeutigen maximalen Varietäten **W** besteht, für die $\mathbf{V} = \mathbf{W} \cap \mathbf{B}$ gilt, wobei **V** eine beliebige Varietät innerhalb von **B** ist. Diese Hierarchie, die Trotter-Weil-Hierarchie gennant wird, ist in Abbildung 2.2 dargestellt und auf der rechten Seite der Abbildung befindet sich die Quantorenalternierungshierarchie, wobei die Level mit den entsprechenden Leveln der Trotter-Weil-Hierarchie auf gleicher Höhe übereinstimmen.

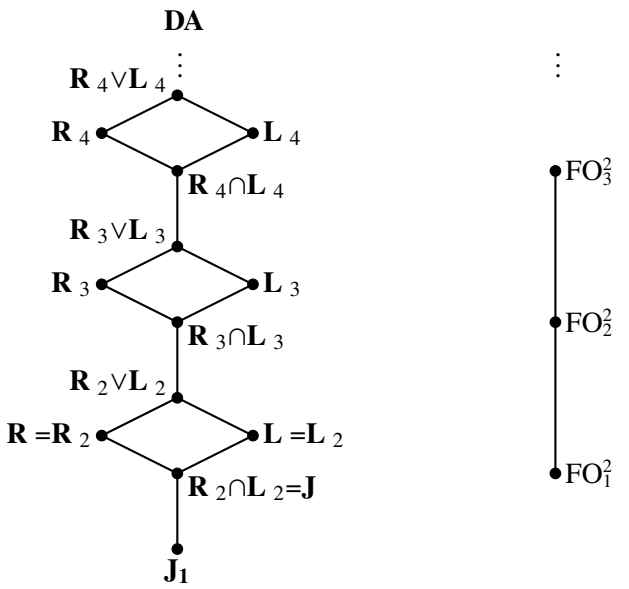



**Abbildung 2.2:** Die Trotter-Weil-Hierarchie.

Hierbei ist $\mathbf{V} \vee \mathbf{W}$ für zwei Varietäten $\mathbf{V}$ und $\mathbf{W}$ definiert als die kleinste Varietät, die $\mathbf{V}$ und $\mathbf{W}$ enthält. Wir interessieren uns also für die Varietäten in Tabelle 2.2 mit $m \geq 2$. Hierbei sind $e_1 = f_1 = 1$, sowie $e_{i+1} = (e_i \dots e_1 z f_1 \dots f_i x_i)^\omega$ und $f_{i+1} = (y_i e_i \dots e_1 z f_1 \dots f_i)^\omega$.

| Symbol | Bezeichnung | Zugehörige $\omega$-Gleichungen |
|---|---|---|
| $\mathbf{R}_m$ | rechte Ecke | $[\![ e_m \dots e_1 z f_1 \dots f_{m-1} = e_m \dots e_1 f_1 \dots f_{m-1} ]\!]$ |
| $\mathbf{L}_m$ | linke Ecke | $[\![ e_{m-1} \dots e_1 z f_1 \dots f_m = e_{m-1} \dots e_1 f_1 \dots f_m ]\!]$ |
| $\mathbf{R}_m \vee \mathbf{L}_m$ | join Level | $[\![ e_m \dots e_1 z f_1 \dots f_m = e_m \dots e_1 f_1 \dots f_m ]\!]$ |
| $\mathbf{R}_m \cap \mathbf{L}_m$ | Schnittlevel | $[\![ e_{m-1} \dots e_1 z f_1 \dots f_{m-1} = e_{m-1} \dots e_1 f_1 \dots f_{m-1} ]\!]$ |

**Tabelle 2.2:** Spezifizierung von Varietäten der Trotter-Weil-Hierarchie.

Die Hierarchie kann auch rekursiv durch Malcev-Produkte mit den Varietäten $\mathbf{K}$ und $\mathbf{D}$ aufgebaut werden. Ein Malcev Produkt zweier Varietäten $\mathbf{V}$ und $\mathbf{W}$ wird mit $\mathbf{V}Ⓜ\mathbf{W}$ bezeichnet. Die hier benötigten Malcev Produkte können am einfachsten über die beiden Kongruenzen $\sim_\mathbf{K}$ und $\sim_\mathbf{D}$ definiert werden. Sei $M$ ein Monoid und $m, n \in M$. Dann gelten die folgenden Definitionen für diese Kongruenzen:

$$m \sim_\mathbf{K} n, \text{ falls } \forall e \in E(M) : em, en \leq_\mathcal{J} e \vee em = en$$

$$m \sim_\mathbf{D} n, \text{ falls } \forall e \in E(M) : me, ne \leq_\mathcal{J} e \vee me = ne$$

Für ein Monoid $M$ und eine Varietät $\mathbf{V}$ gilt dann Folgendes:

$$M \in \mathbf{K}Ⓜ\mathbf{V} \iff M/\sim_K \in \mathbf{V}$$

$$M \in \mathbf{D}Ⓜ\mathbf{V} \iff M/\sim_D \in \mathbf{V}$$

Bei der Trotter-Weil-Hierarchie gilt somit $\mathbf{R}_{m+1} = \mathbf{K}Ⓜ\mathbf{L}_m$ und $\mathbf{L}_{m+1} = \mathbf{D}Ⓜ\mathbf{R}_m$ für $m \geq 2$.

## 2.4 Automaten und Sprachen

Im Folgenden sei $\Sigma$ ein endliches Alphabet und $\Sigma^\omega$ enthalte alle unendlichen Wörter über $\Sigma$.

**Definition 2.9 (Deterministischer Endlicher Automat)** *Ein DEA ist ein Tupel $\mathcal{A} = (Z, \Sigma, \cdot, z_0, F)$ bestehend aus einer Zustandsmenge $Z$, einem endlichen Alphabet $\Sigma$, einer Überführungsfunktion $\cdot : Z \times \Sigma \to Z$, einem Startzustand $q_0 \in Z$ und einer Menge von Endzuständen $F \subseteq Z$.*

Ein Pfad in einem Automaten ist ein Folge $z_{i_1} \cdot \sigma_{i_1} \cdots z_{i_n} \cdot \sigma_{i_n}$ mit $z_{i_j} \in Z$ und $\sigma_{i_j} \in \Sigma$ für $j \in \{1, \ldots, n\}$. Ein Pfad ist akzeptierend, wenn $z_{i_1} = z_0$ und $z_{i_n} \in F$ gilt.

**Definition 2.10 (Reverse deterministischer endlicher Automat)** *Ein reverse DEA ist ein Tupel $\mathcal{A}^\rho = (Z, \Sigma, \cdot^\rho, I, z_f)$ bestehend aus einer Zustandsmenge $Z$, einem endlichen Alphabet $\Sigma$, einer reverse Überführungsfunktion $\cdot^\rho : Z \times \Sigma \to Z$, einer Menge von Startzuständen $I \subseteq Z$ und einem Endzustand $z_f \in Z$.*

Für jeden DEA $\mathcal{A} = (Z, \Sigma, \cdot, z_0, F)$ gibt es einen reverse DEA $\mathcal{A}^\rho = (Z, \Sigma, \cdot^\rho, F, z_0)$, der $\overline{L(\mathcal{A})}$ akzeptiert.

**Definition 2.11 (Nichtdeterministischer Endlicher Automat)** *Ein nichtdeterministischer endlicher Automat (NEA) ist ein Tupel $\mathcal{A} = (Z, \Sigma, \cdot, q_0, F)$ bestehend aus einer Zustandsmenge $Z$, einem endlichen Alphabet $\Sigma$, einer Überführungsrelation $\cdot \subseteq (Z \times \Sigma) \times Z$, einem Startzustand $q_0 \in Z$ und einer Menge von Endzuständen $F \subseteq Z$.*

DEAs und NEAs erkennen nur reguläre Sprachen aus $\Sigma^*$, also Sprachen, deren Wörter alle endliche sind. Für diese Arbeit sind auch $\omega$-reguläre Sprachen relevant, das sind Sprachen aus $\Sigma^\omega$, diese bestehen aus unendlichen Wörtern. Sprachen aus $\Sigma^\omega$ lassen sich durch Büchi-Automaten erkennen. Um alle $\omega$-regulären Sprachen zu erkennen, ist eine Spezialisierung dieser Büchi-Automaten notwendig, die von Carton und Michel entwickelt wurde und nach ihnen benannt ist [3].

**Definition 2.12 (Büchi-Automat)** *Ein Büchi-Automat ist ein Tupel $\mathcal{A} = (Z, \Sigma, \cdot, I, F)$ bestehend aus einer Zustandsmenge $Z$, einem endlichen Alphabet $\Sigma$, einer Überführungsrelation $\cdot \subseteq (Z \times \Sigma) \times Z$, einer Menge von Startzuständen $I \subseteq Z$ und einer Menge von Endzuständen $F \subseteq Z$.*

Ein Pfad in einem Büchi-Automaten wird final genannt, wenn er unendlich oft einen Endzustand enthält und er wird initial genannt, wenn mit einem Startzustand beginnt. Die akzeptierte Sprache $L(\mathcal{A})$ eines Büchi-Automaten $\mathcal{A}$ besteht aus allen unendlichen Worten, die einen Pfad beschriften, der initial und final ist.

**Definition 2.13 (Carton-Michel Automat)** *Ein Carton-Michel Automat (CMA) ist ein Büchi-Automat, bei dem jedes unendliche Wort $w \in \Sigma^\omega$ die Beschriftung genau eines finalen Pfades ist.*

Die akzeptierte Sprache $L(\mathcal{A})$ eines CMA $\mathcal{A}$ besteht aus allen unendlichen Worten, deren finaler Pfad initial ist.

**Definition 2.14 (syntaktische Kongruenz)** *Sei $L \subseteq \Sigma^*$ und $u, v \in \Sigma^*$. Wir setzen $u \equiv_L v$, falls $\forall x, y \in \Sigma^* : xuy \in L \iff xvy \in L$ gilt.*

Das syntaktische Monoid $Synt(L) := \Sigma^*/\equiv_L$ besteht dann aus den Äquivalenzklassen bezüglich dieser Kongruenz mit der Verknüpfung $[u]_L \cdot [v]_L := [uv]_L$ .

Die syntaktische Kongruenz kann nach Arnold auch für Sprachen aus $\Sigma^\omega$ definiert werden [1]. Sei $L \subseteq \Sigma^\omega$ und $u, v \in \Sigma^*$. Wir setzen $u \equiv_L v$, falls Folgendes gilt:

$$\forall x, y, z \in \Sigma^* : \left(xuyz^\omega \in L \iff xvyz^\omega \in L\right) \wedge \left(x(uy)^\omega \in L \iff x(vy)^\omega \in L\right)$$

# 3 Verbotsmuster

In diesem Kapitel wird das Konzept der Verbotsmuster genauer betrachtet. Verbotsmuster werden bereits seit Arbeiten von Stern aus dem Jahr 1985 für Entscheidungsprobleme bei Automaten verwendet. Dieser benutzt Verbotsmuster, ohne diese als Verbotsmuster zu bezeichnen, um unter Anderem zu entscheiden, ob die von einem Automaten erkannte Sprache piecewise-testable ist beziehungsweise dot-depth eins hat, was bedeutet, das sie sich als boolesche Kombination von Sprachen der Form $w_0\Sigma^* w_1\Sigma^* \dots \Sigma^* w_n$ darstellen lässt mit $w_0, \dots, w_n$ als Buchstaben aus $\Sigma$ für piecewise-testable und als Wörter über $\Sigma$ für dot-depth eins [21], [20]. Darauf aufbauend definieren Schmitz, Wagner und Glaßer Entscheidungskriterien unter dem Namen Verbotsmuster für die ersten Level der Straubing-Thérien-Hierarchie und der Dot-depth-Hierarchie [18], [8], [17]. Eine Frage die Stern unbeantwortet lässt, ist die nach der PSPACE-Vollständigkeit von Aperiodizität endlicher Automaten [21]. Diese wird durch Cho und Huynh positiv beantwortet, die ein Vorgehen definieren, um DEAs auf nichttriviale Zyklen zu untersuchen [4]. Eine Übersicht weiterer bisheriger Arbeiten zu Verbotsmustern ist bei Klíma und Polák zu finden [12]. Die Autoren erheben Verbotsmuster von einem Werkzeug zu einem Konzept, indem sie eine Theorie erstellen, anhand der Erkenntnisse über die von Verbotsmustern charakterisierten Klassen gewonnen werden und Eigenschaften und Anwendungsgebiete von Verbotsmustern gefunden werden können.

In dieser Arbeit interessieren wir uns für Verbotsmuster für die Trotter-Weil-Hierarchie. Wie in Kapitel 2 angegeben, kann das Konzept der Varietäten auf Sprachen ausgeweitet werden. Man kann zeigen, dass es zu einer Varietät $\mathbf{V}$ eine korrespondierende Sprachenvarietät $\mathcal{V}$ gibt, für die $L \in \mathcal{V} \iff Synt(L) \in \mathbf{V}$ gilt. Sprachen können also über ihr syntaktisches Monoid in die Trotter-Weil-Hierarchie eingeordnet werden. In Kapitel 4 werden für die Level der Hierarchie Verbotsmuster gefunden, anhand derer entschieden werden kann, ob eine Sprache in diesem Level enthalten ist. Dabei werden endliche und unendliche Wörter behandelt und es werden deshalb Verbotsmuster für DEAs und CMAs benötigt, die hier definiert werden.

**Definition 3.1 (Verbotsmuster)** *Ein Verbotsmuster $M = (V, E, v_s, v_l, v_r)$ ist ein gerichteter Graph, mit Knotenmenge $V$, Kantenmenge $E \subseteq V^2$ und ausgezeichneten Knoten $v_s, v_l, v_r$. Außerdem sei $X$ eine Variablenmenge und $b : E \to X$ eine Beschriftungsfunktion.*

DEAs können Verbotsmuster enthalten. Sei $\mathcal{A} = (Z, \Sigma, \cdot, z_0, F)$ ein DEA und $M$ ein Verbotsmuster. $\mathcal{A}$ enthält $M$, falls Abbildungen $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$ existieren, sodass die folgenden Bedingungen erfüllt sind:

- $\exists p \in \Sigma^* : z_0 \cdot p = h(v_s)$
- $\exists q \in \Sigma^* : h(v_l) \cdot q \in F \iff h(v_r) \cdot q \notin F$
- $\forall (u, v) \in E : h(u) \cdot g(b((u, v))) = h(v)$

Anstatt der Forderung nach einer Beschriftung, mit der von genau einem der Zustände $h(v_l)$ und $h(v_r)$ ein Endzustand erreicht werden kann,würde die Forderung nach $h(v_l) \neq h(v_r)$ genügen, wenn man voraussetzt, dass es sich bei den betrachteten Automaten um Minimalautomaten handelt. Da wir Verbotsmuster auch auf CMAs anwenden wollen, bei denen eine eindeutige Minimierung nicht möglich ist, ist die hier gewählte Alternative aufgrund der dadurch ermöglichten Einheitlichkeit zu bevorzugen.

Der Lesbarkeit wegen notieren wir im Folgenden mit $g(xy)$ für $x, y \in X$ die Verknüpfung von $g(x)$ und $g(y)$.

Beispielhaft wird nun ein Verbotsmuster für die Varietät **DO** gegeben, welche eine Obervarietät von **DA** ist und damit oberhalb der Trotter-Weil-Hierarchie liegt. **DO** hat die charakterisierende Gleichung $[\![(xy)^\omega = (xy)^\omega(yx)^\omega(xy)^\omega]\!]$. Sei $M_{\mathbf{DO}} = (V, E, v_s, v_l, v_r)$ mit $v_s = v_1, v_l = v_1, v_r = v_6$ das Muster, welches in Abbildung 3.1 dargestellt ist. Mit der mit p beschrifteten eingehenden Kante wird $v_s$ markiert und mit den mit q beschrifteten ausgehenden Kanten werden $v_l$ und $v_r$ markiert. Dies soll visualisieren, dass beim Suchen des Musters in einem DEA, $v_s$ auf einen Zustand abgebildet werden soll, der von einem Startzustand aus erreichbar ist und genau einer der Zustände, auf die $v_l$ und $v_r$ abgebildet werden, zu einem Endzustand führen soll bei gleicher Beschriftung.

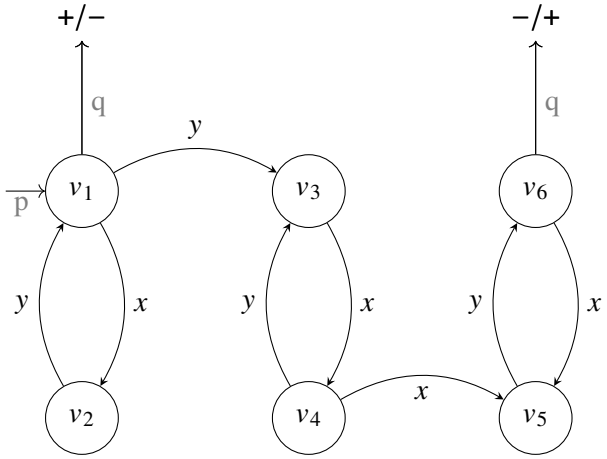



**Abbildung 3.1:** Verbotsmuster $M_{\mathbf{DO}}$ für **DO**.

**Lemma 3.1** *Sei $\mathcal{A}$ ein DEA. Dann ist $M_{\mathbf{DO}}$ in $\mathcal{A}$ genau dann enthalten, wenn $Synt(L(\mathcal{A})) \notin \mathbf{DO}$ gilt.*

*Beweis.* Für die erste Richtung sei $M_{\mathbf{DO}}$ in $\mathcal{A} = (P, \Sigma, \cdot, z_0, F)$ enthalten. Dann existieren Abbildungen $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$, sodass die entsprechenden Bedingungen erfüllt sind. Es folgt $h(v_s) \cdot g((xy)^\omega) = h(v_l)$ und $h(v_s) \cdot g((xy)^\omega(yx)^\omega(xy)^\omega) = h(v_r)$. Weiter folgt $p \cdot g((xy)^\omega) \cdot q \in L(\mathcal{A}) \iff p \cdot g((xy)^\omega(yx)^\omega(xy)^\omega) \cdot q \notin L(\mathcal{A})$. Also sind $g((xy))^\omega$ und $g((xy)^\omega(yx)^\omega(xy)^\omega)$ unterschiedliche Element in $Synt(L(\mathcal{A}))$. Es wurde also eine Variablenbelegung gefunden, welche die $\omega$-Gleichung $[\![(xy)^\omega = (xy)^\omega(yx)^\omega(xy)^\omega]\!]$ nicht erfüllt. Dies führt zu $Synt(L(\mathcal{A})) \notin \mathbf{DO}$.

Für die andere Richtung sei $Synt(L(\mathcal{A})) \notin \mathbf{DO}$. Dann existieren Elemente $\tilde{x}, \tilde{y} \in \Sigma^*$, für die $(\tilde{x}\tilde{y})^\omega \neq (\tilde{x}\tilde{y})^\omega(\tilde{y}\tilde{x})^\omega(\tilde{x}\tilde{y})^\omega$ gilt. Dadurch existieren $p, q \in Synt(L(\mathcal{A}))$ mit $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \cdot q \in F \iff z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega(\tilde{y}\tilde{x})^\omega(\tilde{x}\tilde{y})^\omega \cdot q \notin F$. Da $\omega$ der Idempotenz-Exponent ist, gilt weiter $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \cdot (\tilde{x}\tilde{y})^\omega = z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega$, sowie $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega(\tilde{y}\tilde{x})^\omega(\tilde{y}\tilde{x})^\omega = z_o \cdot p \cdot (\tilde{x}\tilde{y})^\omega(\tilde{y}\tilde{x})^\omega$ und $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega(\tilde{y}\tilde{x})^\omega(\tilde{x}\tilde{y})^\omega(\tilde{x}\tilde{y})^\omega = z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega(\tilde{y}\tilde{x})^\omega(\tilde{x}\tilde{y})^\omega$. Die Abbildungen

$$
\begin{aligned}
h(v_1) &= z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega & g(x) &= (\tilde{x}\tilde{y})^{2\omega-1}\tilde{x} \\
h(v_2) &= h(v_1) \cdot (\tilde{x}\tilde{y})^{\omega-1}\tilde{x} & g(y) &= (\tilde{y}\tilde{x})^\omega\tilde{y} \\
h(v_3) &= h(v_1) \cdot (\tilde{y}\tilde{x})^\omega\tilde{y} \\
h(v_4) &= h(v_1) \cdot (\tilde{y}\tilde{x})^\omega \\
h(v_5) &= h(v_4) \cdot (\tilde{x}\tilde{y})^{2\omega-1}\tilde{x} \\
h(v_6) &= h(v_5) \cdot \tilde{y}
\end{aligned}
$$

beweisen nun das Enthaltensein von $M_{\mathbf{DO}}$ in $\mathcal{A}$. □

**Definition 3.2 (Reverse Verbotsmuster)** *Sei $M = (V, E, v_s, v_l, v_r)$ ein Verbotsmuster. Dann ist $M^\rho = (V, E^\rho, v_s, v_l, v_r)$ mit $E^\rho = \{(v, u) \mid (u, v) \in E\}$ ein reverse Verbotsmuster.*

Reverse DEAs können reverse Verbotsmuster enthalten. Hierbei gilt: Sei $\mathcal{A}^\rho = (Z, \Sigma, \cdot^\rho, I, z_f)$ ein reverse DEA und $M^\rho$ ein reverse Verbotsmuster. $\mathcal{A}^\rho$ enthält $M^\rho$, falls Abbildungen $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}^\rho))$ existieren, sodass die folgenden Bedingungen erfüllt sind:

- $\exists q \in \Sigma^* : z_f \cdot^\rho q = h(v_s)$
- $\exists p \in \Sigma^* : h(v_l) \cdot^\rho p \in I \iff h(v_r) \cdot^\rho p \notin I$
- $\forall (u, v) \in E^\rho : h(u) \cdot^\rho g(b((u, v))) = h(v)$

Ein reverse DEA enthält also genau dann ein reverse Verbotsmuster, wenn der entsprechende DEA das entsprechende Verbotsmuster enthält.

Auch CMAs können reverse Verbotsmuster enthalten. Sei $\mathcal{A} = (Z, \Sigma, \cdot, I, F)$ ein CMA und $M^\rho$ ein reverse Verbotsmuster. Sei weiter $\cdot^\rho$ die Umkehrrelation von $\cdot$. $\mathcal{A}$ enthält $M^\rho$, falls Abbildungen $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$ existieren, sodass die folgenden Bedingungen erfüllt sind:

- $\exists q \in \Sigma^\omega : h(v_s) \cdot q \in F$
- $\exists p \in \Sigma^* : h(v_l) \cdot^\rho p \in I \iff h(v_r) \cdot^\rho p \notin I$
- $\forall (u, v) \in E^\rho : h(u) \cdot (b((u, v))) = h(v)$

# 4 Verbotsmuster für die Trotter-Weil-Hierarchie

In diesem Kapitel werden Verbotsmuster erarbeitet, die auf die Mitgliedschaft in den Varietäten der Trotter-Weil-Hierarchie schließen lassen. Dabei betrachten wir zuerst endliche Wörter und anschließend unendliche.

## 4.1 Endliche Wörter

Wir beginnen mit den Varietäten an den beiden Enden der Hierarchie, für die wir explizite Verbotsmuster definieren und geben darauffolgend ein allgemeines Schema für die Verbotsmuster der restlichen Hierarchie an. Als erstes betrachten wir **DA**.

### 4.1.1 Varietät DA

Wie in Tabelle 2.1 aufgezeigt ist, gibt es drei $\omega$-Gleichungen für **DA**. Für jede dieser Gleichungen wird nachfolgend ein passendes Verbotsmuster angegeben.

Zuerst betrachten wir die Gleichung $[\![(xy)^\omega = (xy)^\omega x (xy)^\omega]\!]$. Dafür definieren wir das Muster $M_{\mathbf{DA1}} = (V, E, v_s, v_l, v_r)$ mit $v_s = v_1, v_l = v_1, v_r = v_4$, das in Abbildung 4.1 dargestellt ist.

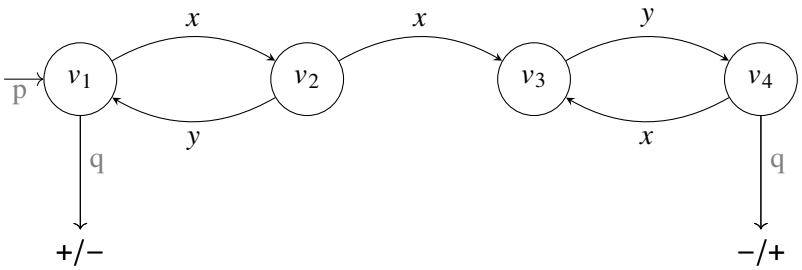



**Abbildung 4.1:** Verbotsmuster $M_{\mathbf{DA1}}$ für **DA**.

**Lemma 4.1** *Sei $\mathcal{A}$ ein DEA. Dann ist $M_{\mathbf{DA1}}$ in $\mathcal{A}$ genau dann enthalten, wenn $Synt(L(\mathcal{A})) \notin \mathbf{DA}$ gilt.*

*Beweis.* Für die erste Richtung sei $M_{\mathbf{DA1}}$ in $\mathcal{A} = (P, \Sigma, \cdot, z_0, F)$ enthalten. Dann existieren Abbildungen $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$, sodass die entsprechenden Bedingungen erfüllt sind. Es folgt $h(v_s) \cdot g((xy)^\omega) = h(v_l)$ und $h(v_s) \cdot g((xy)^\omega x (xy)^\omega) = h(v_r)$. Weiter folgt:

$$p \cdot g((xy)^\omega) \cdot q \in L(\mathcal{A}) \iff p \cdot g((xy)^\omega x (xy)^\omega) \cdot q \notin L(\mathcal{A})$$

Also sind $g((xy)^\omega)$ und $g((xy)^\omega x(xy)^\omega)$ unterschiedliche Element in $Synt(L(\mathcal{A}))$. Es wurde also eine Variablenbelegung gefunden, welche die $\omega$-Gleichung $[\![(xy)^\omega = (xy)^\omega x(xy)^\omega]\!]$ nicht erfüllt. Dies führt zu $Synt(L(\mathcal{A})) \notin \mathbf{DA}$.

Für die andere Richtung sei $Synt(L(\mathcal{A})) \notin \mathbf{DA}$. Dann existieren Elemente $\tilde{x}, \tilde{y} \in Synt(L(\mathcal{A}))$, für die $(\tilde{x}\tilde{y})^\omega \neq (\tilde{x}\tilde{y})^\omega \tilde{x} (\tilde{x}\tilde{y})^\omega$ gilt. Dadurch existieren $p, q \in \Sigma^*$ mit:

$$z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \cdot q \in F \iff z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \tilde{x} (\tilde{x}\tilde{y})^\omega \cdot q \notin F$$

Da $\omega$ der Idempotenz-Exponent ist, gilt weiter $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \cdot (\tilde{x}\tilde{y})^\omega = z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega$ und $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \tilde{x} (\tilde{x}\tilde{y})^\omega = z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \tilde{x} (\tilde{x}\tilde{y})^\omega \cdot (\tilde{x}\tilde{y})^\omega$. Die Abbildungen

$$\begin{aligned} h(v_1) &= z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega & g(x) &= (\tilde{x}\tilde{y})^\omega \tilde{x} \\ h(v_2) &= h(v_1) \cdot \tilde{x} & g(y) &= \tilde{y}(\tilde{x}\tilde{y})^{\omega-1} \\ h(v_3) &= h(v_2) \cdot (\tilde{x}\tilde{y})^\omega \tilde{x} \\ h(v_4) &= h(v_3) \cdot \tilde{y}(\tilde{x}\tilde{y})^{\omega-1} \end{aligned}$$

beweisen nun das Enthaltensein von $M_{\mathbf{DA1}}$ in $\mathcal{A}$. □

Die betrachtete Gleichung für **DA** legt ein anderes Muster nahe, welches die Varietät jedoch nicht korrekt charakterisiert. Abbildung 4.2 zeigt das Muster, welches aussieht, als würde es zu $[\![(xy)^\omega = (xy)^\omega x(xy)^\omega]\!]$=**DA** gehören.

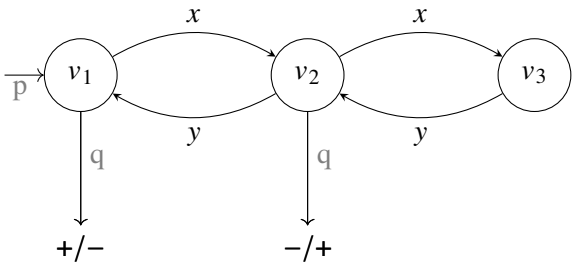



**Abbildung 4.2:** Verbotsmuster $M_{\text{not}\mathbf{DA}}$.

**Lemma 4.2** *Sei $\mathcal{A}$ ein DEA. Das Muster $M_{not\mathbf{DA}}$ erfüllt nicht die Äquivalenz: $M_{not\mathbf{DA}}$ ist in $\mathcal{A}$ enthalten $\iff$ $Synt(L(\mathcal{A})) \notin \mathbf{DA}$.*

*Beweis.* Angenommen, die Äquivalenz würde gelten. Wir führen dies mittels eines Gegenbeispiels zum Widerspruch. Wir betrachten die Sprache $L = (ab)^*$, welche vom DEA $\mathcal{A}$ in Abbildung 4.3 erkannt wird. Es gilt $L \in \mathbf{A} \setminus \mathbf{DA}$ und wir werden zeigen, dass $M$ dennoch nicht in $\mathcal{A}$ enthalten ist.

Damit $M_{\text{not}\mathbf{DA}}$ in $\mathcal{A}$ enthalten sein kann, muss es entsprechende Abbildungen $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$ geben. Für diese muss $h(v_1) \neq h(v_2)$ gelten und es muss Pfade von $h(v_1)$ zu $h(v_2)$ und umgekehrt geben. Dadurch folgt $\big(h(v_1) = z_1 \wedge h(v_2) = z_2\big) \vee \big(h(v_1) = z_2 \wedge h(v_2) = z_1\big)$. Weiter kann $h(v_3) = z_f$ nicht gelten, da es sonst keinen Pfad von $h(v_3)$ zu $h(v_2)$ gäbe. Außerdem kann $h(v_3) = h(v_2)$ nicht gelten, da es sonst keine gleichbeschrifteten Pfade von $h(v_1)$ zu $h(v_2)$ und von $h(v_2)$ zu $h(v_3)$ gäbe. Zuletzt kann $h(v_3) = h(v_1)$ aus dem selben Grund nicht gelten. Die Abbildungen $h$ und $g$ können also nicht existieren. □

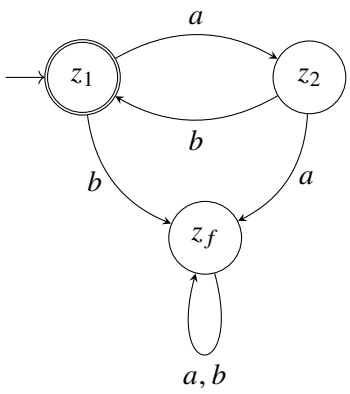



**Abbildung 4.3:** DEA $\mathcal{A}$ mit $L(\mathcal{A}) = (ab)^*$.

Die beiden anderen $\omega$-Gleichungen für **DA** aus Tabelle 2.1, $[\![(xy)^\omega = (xy)^\omega y(xy)^\omega]\!]$ und $[\![(xyz)^\omega = (xyz)^\omega y(xyz)^\omega]\!]$ werden durch die Muster $M_{\mathbf{DA2}}$ beziehungsweise $M_{\mathbf{DA3}}$ charakterisiert, die in Abbildung 4.4 beziehungsweise Abbildung 4.5 dargestellt sind.

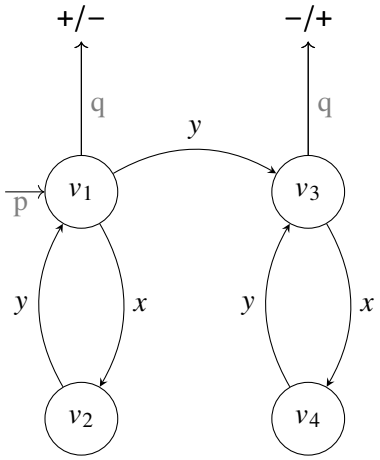



**Abbildung 4.4:** Verbotsmuster $M_{\mathbf{DA2}}$ für **DA**.

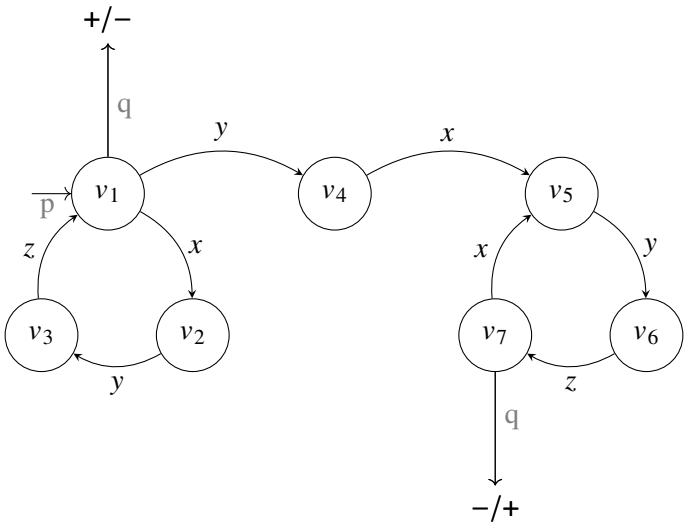



**Abbildung 4.5:** Verbotsmuster $M_{\mathbf{DA3}}$ für **DA**.

Die Beweise, dass es sich bei $M_{\mathbf{DA2}}$ und $M_{\mathbf{DA3}}$ tatsächlich um Verbotsmuster für **DA** handelt, funktionieren analog zum Beweis für Lemma 4.1. Im Fall von $M_{\mathbf{DA2}}$ verwendet man die folgenden Abbildungen:

$$h(v_1) = z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \qquad g(x) = (\tilde{x}\tilde{y})^{\omega-1}\tilde{x}$$
$$h(v_2) = h(v_1) \cdot (\tilde{x}\tilde{y})^{\omega-1}\tilde{x} \qquad g(y) = \tilde{y}(\tilde{x}\tilde{y})^\omega$$
$$h(v_3) = h(v_1) \cdot \tilde{y}(\tilde{x}\tilde{y})^\omega$$
$$h(v_4) = h(v_3) \cdot (\tilde{x}\tilde{y})^{\omega-1}\tilde{x}$$

Im Fall von $M_{\mathbf{DA3}}$ verwendet man die folgenden Abbildungen:

$$h(v_1) = z_0 \cdot p \cdot (\tilde{x}\tilde{y}\tilde{z})^\omega \qquad g(x) = (\tilde{x}\tilde{y}\tilde{z})^\omega\tilde{x}$$
$$h(v_2) = h(v_1) \cdot \tilde{x} \qquad g(y) = \tilde{y}$$
$$h(v_3) = h(v_2) \cdot \tilde{y} \qquad g(z) = \tilde{z}(\tilde{x}\tilde{y}\tilde{z})^{\omega-1}$$
$$h(v_4) = h(v_1) \cdot \tilde{y}$$
$$h(v_5) = h(v_4) \cdot (\tilde{x}\tilde{y}\tilde{z})^\omega\tilde{x}$$
$$h(v_6) = h(v_5) \cdot \tilde{y}$$
$$h(v_7) = h(v_6) \cdot \tilde{z}(\tilde{x}\tilde{y}\tilde{z})^{\omega-1}$$

### 4.1.2 Varietät J

Wie in Tabelle 2.1 angegeben, ist $[\![(xy)^\omega x(ts)^\omega = (xy)^\omega s(ts)^\omega]\!]$ die $\omega$-Gleichung für **J**. Als Verbotsmuster für diese definieren wir $M_{\mathbf{J}} = (V, E, v_s, v_l, v_r)$ mit $v_s = v_1, v_l = v_4, v_r = v_5$ entsprechend Abbildung 4.6.

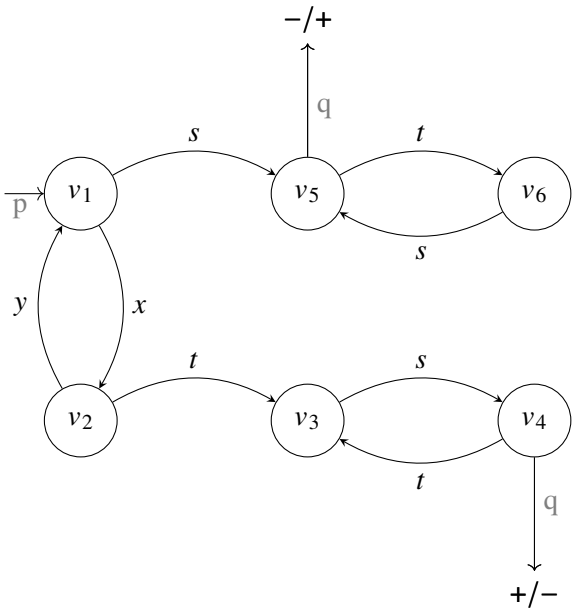



**Abbildung 4.6:** Verbotsmuster $M_{\mathbf{J}}$ für $\mathbf{J}$.

**Lemma 4.3** *Sei $\mathcal{A}$ ein DEA. Dann ist $M_{\mathbf{J}}$ in $\mathcal{A}$ genau dann enthalten, wenn $Synt(L(\mathcal{A})) \notin \mathbf{J}$ gilt.*

*Beweis.* Für die erste Richtung sei $M_{\mathbf{J}}$ in $\mathcal{A} = (P, \Sigma, \cdot, z_0, F)$ enthalten. Dann existieren Abbildungen $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$, sodass die entsprechenden Bedingungen erfüllt sind. Es folgt $h(v_s) \cdot g((xy)^\omega x(ts)^\omega) = h(v_l)$ und $h(v_s) \cdot g((xy)^\omega s(ts)^\omega) = h(v_r)$. Weiter folgt:

$$p \cdot g((xy)^\omega x(ts)^\omega) \cdot q \in L(\mathcal{A}) \iff p \cdot g((xy)^\omega s(ts)^\omega) \cdot q \notin L(\mathcal{A})$$

Also sind $g((xy)^\omega x(ts)^\omega)$ und $g((xy)^\omega s(ts)^\omega)$ unterschiedliche Element in $Synt(L(\mathcal{A}))$. Es wurde also eine Variablenbelegung gefunden, welche die $\omega$-Gleichung $[\![(xy)^\omega x(ts)^\omega = (xy)^\omega s(ts)^\omega]\!]$ nicht erfüllt. Dies führt zu $Synt(L(\mathcal{A})) \notin \mathbf{J}$.

Für die andere Richtung sei $Synt(L(\mathcal{A})) \notin \mathbf{J}$. Dann existieren Elemente $\tilde{x}, \tilde{y}, \tilde{t}, \tilde{s} \in Synt(L(\mathcal{A}))$, für die $(\tilde{x}\tilde{y})^\omega \tilde{x}(\tilde{t}\tilde{s})^\omega \neq (\tilde{x}\tilde{y})^\omega \tilde{s}(\tilde{t}\tilde{s})^\omega$ gilt. Dadurch existieren $p, q \in \Sigma^*$ mit:

$$z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \tilde{x}(\tilde{t}\tilde{s})^\omega \cdot q \in F \iff z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \tilde{s}(\tilde{t}\tilde{s})^\omega \cdot q \notin F$$

Da $\omega$ der Idempotenz-Exponent ist, gilt weiter $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \cdot (\tilde{x}\tilde{y})^\omega = z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega$, sowie $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega x(\tilde{t}\tilde{s})^\omega (\tilde{t}\tilde{s})^\omega = z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega x(\tilde{t}\tilde{s})^\omega$ und $z_0 \cdot (\tilde{x}\tilde{y})^\omega (\tilde{s}\tilde{t})^\omega (\tilde{s}\tilde{t})^\omega = z_0 \cdot (\tilde{x}\tilde{y})^\omega (\tilde{s}\tilde{t})^\omega$. Die Abbildungen

$$\begin{aligned}
h(v_1) &= z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega & g(x) &= \tilde{x} \\
h(v_2) &= h(v_1) \cdot \tilde{x} & g(y) &= \tilde{y}(\tilde{x}\tilde{y})^{\omega-1} \\
h(v_3) &= h(v_2) \cdot (\tilde{t}\tilde{s})^{2\omega-1}\tilde{t} & g(s) &= (\tilde{s}\tilde{t})^\omega \tilde{s} \\
h(v_4) &= h(v_2) \cdot (\tilde{t}\tilde{s})^\omega & g(t) &= (\tilde{t}\tilde{s})^{2\omega-1}\tilde{t} \\
h(v_5) &= h(v_1) \cdot (\tilde{s}\tilde{t})^\omega \tilde{s} & & \\
h(v_6) &= h(v_1) \cdot (\tilde{s}\tilde{t})^\omega & &
\end{aligned}$$

beweisen nun das Enthaltensein von $M_{\mathbf{J}}$ in $\mathcal{A}$. □

### 4.1.3 Varietät R

In Tabelle 2.1 ist zu sehen, dass es zwei $\omega$-Gleichungen für **R** gibt. Für jede dieser Gleichungen wird nachfolgend ein passendes Verbotsmuster angegeben.

Wir beginnen mit der Gleichung $[\![(xy)^\omega = (xy)^\omega x]\!]$. Wir definieren das Muster $M_{\mathbf{R1}} = (V, E, v_s, v_l, v_r)$ mit $v_s = v_1, v_l = v_1, v_r = v_2$, das in Abbildung 4.7 dargestellt ist.

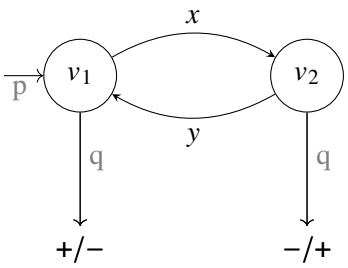



**Abbildung 4.7:** Verbotsmuster $M_{\mathbf{R1}}$ für **R**.

**Lemma 4.4** *Sei $\mathcal{A}$ ein DEA. Dann ist $M_{\mathbf{R1}}$ in $\mathcal{A}$ genau dann enthalten, wenn $Synt(L(\mathcal{A})) \notin \mathbf{R}$ gilt.*

*Beweis.* Für die erste Richtung sei $M_{\mathbf{R1}}$ in $\mathcal{A} = (P, \Sigma, \cdot, z_0, F)$ enthalten. Dann existieren Abbildungen $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$, sodass die entsprechenden Bedingungen erfüllt sind. Es folgt $h(v_s) \cdot g((xy)^\omega) = h(v_l)$ und $h(v_s) \cdot g((xy)^\omega x) = h(v_r)$. Weiter folgt:

$$p \cdot g((xy)^\omega) \cdot q \in L(\mathcal{A}) \iff p \cdot g((xy)^\omega x) \cdot q \notin L(\mathcal{A})$$

Also sind $g((xy)^\omega)$ und $g((xy)^\omega x)$ unterschiedliche Element in $Synt(L(\mathcal{A}))$. Es wurde also eine Variablenbelegung gefunden, welche die $\omega$-Gleichung $[\![(xy)^\omega = (xy)^\omega x]\!]$ nicht erfüllt. Dies führt zu $Synt(L(\mathcal{A})) \notin \mathbf{R}$.

Für die andere Richtung sei $Synt(L(\mathcal{A})) \notin \mathbf{R}$. Dann existieren Elemente $\tilde{x}, \tilde{y} \in Synt(L(\mathcal{A}))$, für die $(\tilde{x}\tilde{y})^\omega \neq (\tilde{x}\tilde{y})^\omega \tilde{x}$ gilt. Dadurch existieren $p, q \in \Sigma^*$ mit:

$$z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \cdot q \in F \iff z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \tilde{x} \cdot q \notin F$$

Da $\omega$ der Idempotenz-Exponent ist, gilt weiter $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \cdot (\tilde{x}\tilde{y})^\omega = z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega$. Die Abbildungen

$$h(v_1) = z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \qquad g(x) = \tilde{x}$$
$$h(v_2) = h(v_1) \cdot \tilde{x} \qquad g(y) = \tilde{y}(\tilde{x}\tilde{y})^{\omega-1}$$

beweisen nun das Enthaltensein von $M_{\mathbf{R1}}$ in $\mathcal{A}$. □

Analog dazu kann man beweisen, dass es sich bei dem Verbotsmuster $M_{\mathbf{R2}}$, das in Abbildung 4.8 dargestellt ist, um das korrekte Muster für die zweite $\omega$-Gleichung $\mathbf{R} = [\![(xyz)^\omega = (xyz)^\omega y]\!]$ handelt. Für den Beweis verwendet man die folgenden Abbildungen:

$$h(v_1) = z_0 \cdot p \cdot (\tilde{x}\tilde{y}\tilde{z})^\omega \qquad g(x) = \tilde{x}$$
$$h(v_2) = h(v_1) \cdot \tilde{x} \qquad g(y) = \tilde{y}$$
$$h(v_3) = h(v_2) \cdot \tilde{y} \qquad g(z) = \tilde{z}(\tilde{x}\tilde{y}\tilde{z})^{\omega-1}$$
$$h(v_4) = h(v_1) \cdot \tilde{y}$$

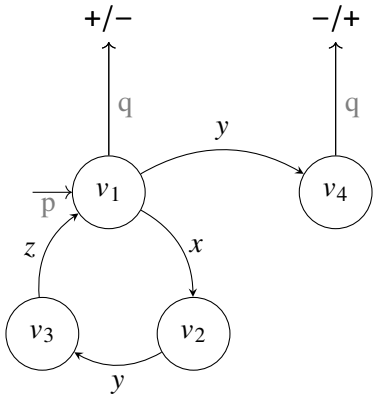



**Abbildung 4.8:** Verbotsmuster $M_{\mathbf{R2}}$ für $\mathbf{R}$.

### 4.1.4 Varietät L

Als nächstes betrachten wir die beiden Gleichungen aus Tabelle 2.1 für $\mathbf{L}$. Für beide Gleichungen wird im Folgenden ein passendes Verbotsmuster angegeben.

Wir beginnen mit der Gleichung $[\![(xy)^\omega = y(xy)^\omega]\!]$. Wir definieren das Muster $M_{\mathbf{L1}} = (V, E, v_s, v_l, v_r)$ mit $v_s = v_1, v_l = v_3, v_r = v_5$, das in Abbildung 4.9 dargestellt ist.

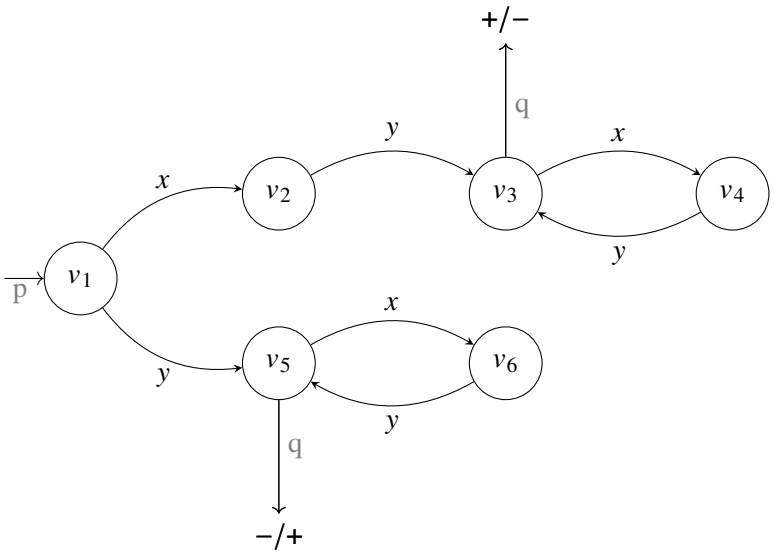



**Abbildung 4.9:** Verbotsmuster $M_{\mathbf{L1}}$ für $\mathbf{L}$.

**Lemma 4.5** *Sei $\mathcal{A}$ ein DEA. Dann ist $M_{\mathbf{L1}}$ in $\mathcal{A}$ genau dann enthalten, wenn $Synt(L(\mathcal{A})) \notin \mathbf{L}$ gilt.*

*Beweis.* Für die erste Richtung sei $M_{\mathbf{L1}}$ in $\mathcal{A} = (P, \Sigma, \cdot, z_0, F)$ enthalten. Dann existieren Abbildungen $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$, sodass die entsprechenden Bedingungen erfüllt sind. Es folgt $h(v_s) \cdot g((xy)^\omega) = h(v_l)$ und $h(v_s) \cdot g(y(xy)^\omega) = h(v_r)$. Weiter folgt:

$$p \cdot g((xy)^\omega) \cdot q \in L(\mathcal{A}) \iff p \cdot g(y(xy)^\omega) \cdot q \notin L(\mathcal{A})$$

Also sind $g((xy)^\omega)$ und $g(y(xy)^\omega)$ unterschiedliche Element in $Synt(L(\mathcal{A}))$. Es wurde also eine Variablenbelegung gefunden, welche die $\omega$-Gleichung $[\![(xy)^\omega = y(xy)^\omega]\!]$ nicht erfüllt. Dies führt zu $Synt(L(\mathcal{A})) \notin \mathbf{L}$.

Für die andere Richtung sei $Synt(L(\mathcal{A})) \notin \mathbf{L}$. Dann existieren Elemente $\tilde{x}, \tilde{y} \in Synt(L(\mathcal{A}))$, für die $(\tilde{x}\tilde{y})^\omega \neq \tilde{y}(\tilde{x}\tilde{y})^\omega$ gilt. Dadurch existieren $p, q \in \Sigma^*$ mit:

$$z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \cdot q \in F \iff z_0 \cdot p \cdot \tilde{y}(\tilde{x}\tilde{y})^\omega \cdot q \notin F$$

Da $\omega$ der Idempotenz-Exponent ist, gilt weiter $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega \cdot (\tilde{x}\tilde{y})^\omega = z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega$, sowie $z_0 \cdot p \cdot (\tilde{y}\tilde{x})^\omega \cdot (\tilde{y}\tilde{x})^\omega = z_0 \cdot p \cdot (\tilde{y}\tilde{x})^\omega$. Die Abbildungen

$$\begin{aligned}
h(v_1) &= z_0 \cdot p & g(x) &= (\tilde{x}\tilde{y})^{\omega-1}\tilde{x} \\
h(v_2) &= h(v_1) \cdot (\tilde{x}\tilde{y})^{\omega-1}\tilde{x} & g(y) &= (\tilde{y}\tilde{x})^\omega\tilde{y} \\
h(v_3) &= h(v_2) \cdot \tilde{y} \\
h(v_4) &= h(v_3) \cdot (\tilde{x}\tilde{y})^{\omega-1}\tilde{x} \\
h(v_5) &= h(v_1) \cdot (\tilde{y}\tilde{x})^\omega\tilde{y} \\
h(v_6) &= h(v_1) \cdot (\tilde{y}\tilde{x})^\omega
\end{aligned}$$

beweisen nun das Enthaltensein von $M_{\mathbf{L1}}$ in $\mathcal{A}$. □

Analog dazu kann mit den folgenden Gleichungen beweisen werden, dass es sich bei dem Verbotsmuster $M_{\mathbf{L2}}$, das in Abbildung 4.10 dargestellt ist, um das korrekte Muster für die zweite $\omega$-Gleichung $\mathbf{L} = [\![ (xyz)^\omega = y(xyz)^\omega ]\!]$ handelt.

$$
\begin{aligned}
h(v_1) &= z_0 \cdot p & g(x) &= (\tilde{x}\tilde{y}\tilde{z})^{2\omega-1}\tilde{x} \\
h(v_2) &= h(v_1) \cdot (\tilde{x}\tilde{y}\tilde{z})^{2\omega-1}\tilde{x} & g(y) &= \tilde{y} \\
h(v_3) &= h(v_2) \cdot \tilde{y} & g(z) &= \tilde{z} \\
h(v_4) &= h(v_1) \cdot \tilde{z} \\
h(v_5) &= h(v_1) \cdot \tilde{y} \\
h(v_6) &= h(v_5) \cdot (\tilde{x}\tilde{y}\tilde{z})^{2\omega-1}\tilde{x} \\
h(v_7) &= h(v_6) \cdot \tilde{y} \\
h(v_8) &= h(v_7) \cdot \tilde{z}
\end{aligned}
$$

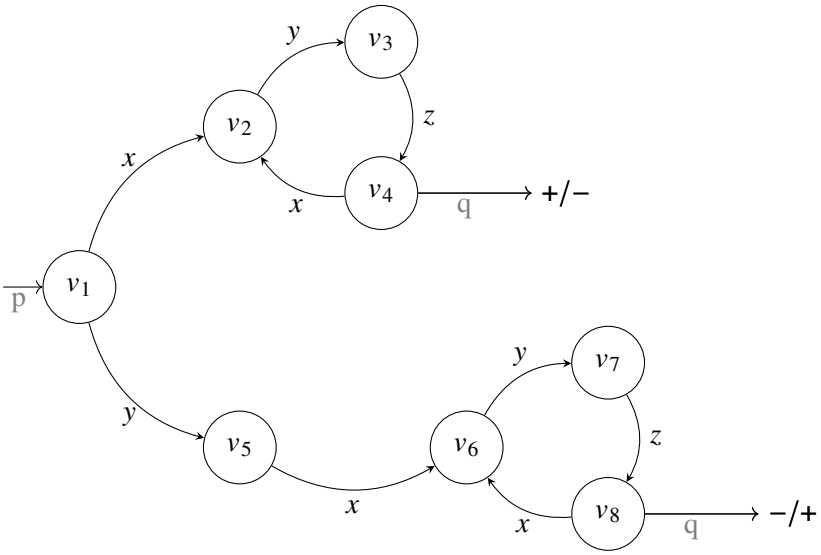



**Abbildung 4.10:** Verbotsmuster $M_{\mathbf{L2}}$ für $\mathbf{L}$.

### 4.1.5 Innere Varietäten der Trotter-Weil-Hierarchie

Schließlich wird ein Schema für die Verbotsmuster der verbleibenden Level der Hierarchie vorgestellt. Die Verbotsmuster in diesem Abschnitt sind in ähnlicher Form bei Rupp zu finden, wobei dort Minimalität der betrachteten DEAs gefordert wird [16]. Die $\omega$-Gleichungen in Tabelle 2.2 legen eine rekursive Musterdefinition nahe. Wir starten mit Grundmustern für $\mathbf{R}_m$, $\mathbf{L}_m$, $\mathbf{R}_m \cap \mathbf{L}_m$ und $\mathbf{R}_m \vee \mathbf{L}_m$ und bauen diese anschließend zu den finalen Verbotsmustern aus. Die Grundmuster seien so definiert, wie in Abbildung 4.11 dargestellt.

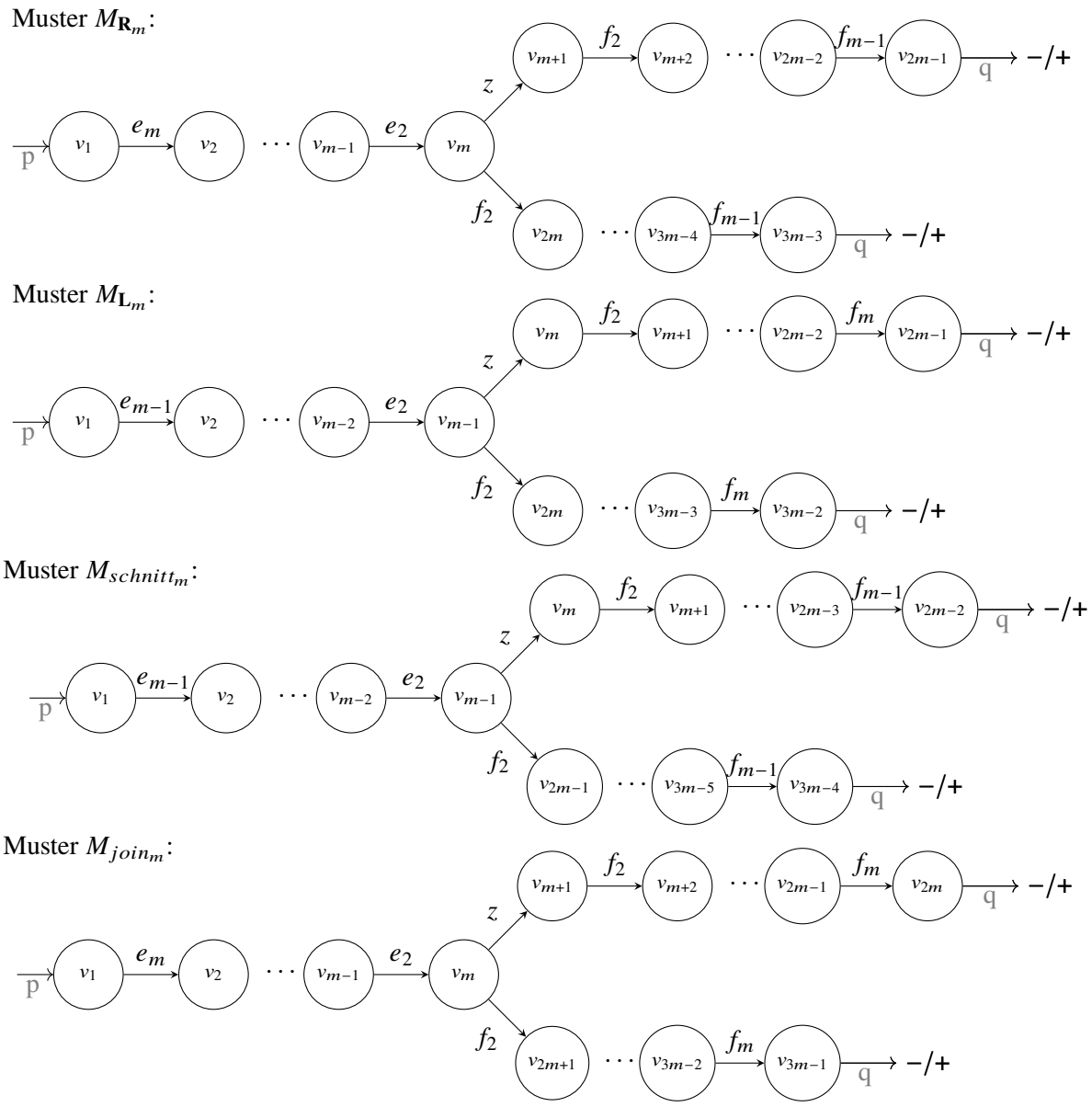
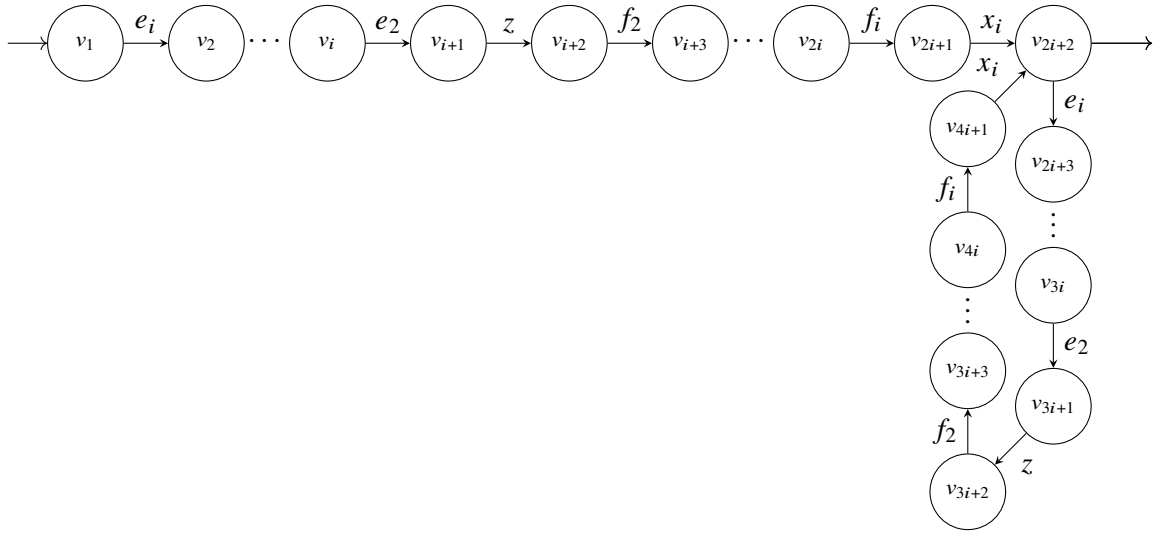



**Abbildung 4.11:** Grundmuster $M_{\mathbf{R}_m}$, $M_{\mathbf{L}_m}$, $M_{schnitt_m}$ und $M_{join_m}$ als Grundlage für die Verbotsmuster der Trotter-Weil-Hierarchie.

Um aus den Grundmustern die tatsächlichen Verbotsmuster zu erhalten, müssen die Kanten mit Beschriftungen $e_i$ oder $f_i$ mit $2 \leq i \leq m$ noch ersetzt werden. Wir definieren die Muster in Abbildung 4.12 und Abbildung 4.13 für diese Kanten.

Muster $M_{e_{i+1}}$:




Muster $M_{f_{i+1}}$:

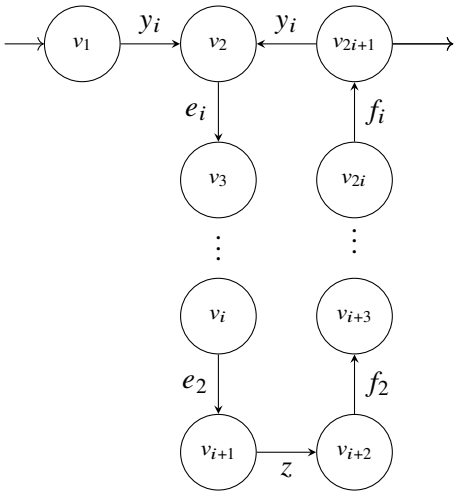
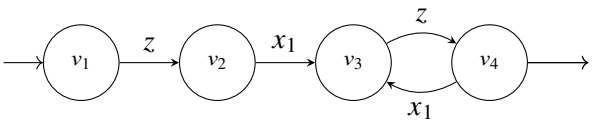



**Abbildung 4.12:** Kantenmuster $M_{e_{i+1}}$ und $M_{f_{i+1}}$ zum Entwickeln der Verbotsmuster der Trotter-Weil-Hierarchie.

Muster $M_{e_2}$:




Muster $M_{f_2}$:

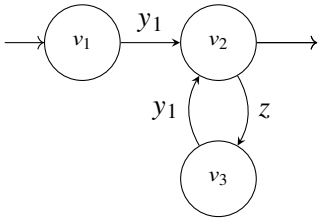



**Abbildung 4.13:** Kantenmuster $M_{e_2}$ und $M_{f_2}$ zum Entwickeln der Verbotsmuster der Trotter-Weil-Hierarchie.

Um nun das Verbotsmuster für eine Varietät der Trotter-Weil-Hierarchie zu erhalten, startet man mit dem entsprechenden Grundmuster und verfeinert dieses schrittweise bis keine Kanten mit $e$- oder $f$-Beschriftungen mehr enthalten sind. Sei $i+1$ der höchste vorkommenden Index einer $e$ oder $f$ Variablen. Dann startet man damit, alle existierenden Kanten mit der Beschriftung $e_{i+1}$ durch das Muster $M_{e_{i+1}}$ und alle existierenden Kanten mit der Beschriftung $f_{i+1}$ durch das Muster $M_{f_{i+1}}$ zu ersetzen. Der Knoten in dem eingsetzten Muster, der mit einer eingehenden Kante markiert ist, fällt hierbei mit dem Startknoten der ersetzten Kante zusammen und der Knoten in dem eingesetzten Muster, der mit einer ausgehenden Kante markiert ist, fällt mit dem Endknoten der ersetzten Kante zusammen. Anschließend wird das Vorgehen mit dem nächstkleineren Index wiederholt. Die Rekursion endet bei $i = 2$. Das entstandene Verbotsmuster enthält nun nur noch Variablen aus der Menge $\{z, x_1, \ldots, x_m, y_1, \ldots, y_m\}$. Für einen Beweis, dass diese Verbotsmuster die Level der Trotter-Weil-Hierarchie charakterisieren, siehe [16].

## 4.2 Unendliche Wörter

In diesem Abschnitt werden Verbotsmuster für Sprachen aus $\Sigma^\omega$ gebildet. Es werden hierbei CMAs verwendet, eine co-deterministische und co-vollständige Version von Büchi-Automaten. Um Varietäten hierbei korrekt zu charakterisieren, werden je zwei Verbotsmuster benötigt, die das endliche und unendliche Verhalten getrennt betrachten. Die zweigeteilte Definition der syntaktischen Kongruenz bei Sprachen mit unendlichen Wörtern legt die folgende Aufteilung in zwei Kongruenzen nahe [10].

**Definition 4.1 (fin-syntaktische Kongruenz)** *Sei $L \subseteq \Sigma^\omega$ und $u, v \in \Sigma^*$. Wir setzen $u \equiv_{fin} v$, falls $\forall x, y, z \in \Sigma^* : xuyz^\omega \in L \iff xvyz^\omega \in L$ gilt.*

**Definition 4.2 (inf-syntaktische Kongruenz)** *Sei $L \subseteq \Sigma^\omega$ und $u, v \in \Sigma^*$. Wir setzen $u \equiv_{inf} v$, falls $\forall x, y \in \Sigma^* : x(uy)^\omega \in L \iff x(vy)^\omega \in L$ gilt.*

Das fin-syntaktische und inf-syntaktische Monoid $Synt_{fin}(L(\mathcal{A})) := \Sigma^*/\equiv_{fin}$ beziehungsweise $Synt_{inf}(L(\mathcal{A})) := \Sigma^*/\equiv_{inf}$ besteht dann aus den entsprechenden Äquivalenzklassen. Nun sollen Kriterien gefunden werden, um für Varietäten $\mathbf{V}$ und CMAs $\mathcal{A}$ zu entscheiden, ob $Synt_{fin}(L(\mathcal{A})), Synt_{inf}(L(\mathcal{A})) \in \mathbf{V}$ gilt, da genau dann auch $Synt(L(\mathcal{A})) \in \mathbf{V}$ folgt. Das folgende Lemma von Henriksson und Kufleitner generalisiert diesen Zusammenhang.

**Lemma 4.6 (Henriksson und Kufleitner [10])** *Sei $\mathbf{V}$ eine Varietät mit $\mathbf{J}_+ \subseteq \mathbf{V} \subseteq \mathbf{DA}$. Weiter sei $L \subseteq \Sigma^\omega$ eine Sprache. Dann gilt $Synt(L(\mathcal{A})) \in \mathbf{V}$ genau dann, wenn $Synt_{fin}(L(\mathcal{A})) \in \mathbf{V}$ und $Synt_{inf}(L(\mathcal{A})) \in \mathbf{DA}$ gilt.*

Es genügt also, die Mitgliedschaft des fin-syntaktischen Monoids in der gewünschten Varietät und des inf-syntaktischen Monoids in **DA** zu zeigen. Anhand dessen werden nun Verbotsmuster für die Trotter-Weil-Hierarchie Hierarchie für fin- und inf-syntaktische Monoide angegeben. Dafür beginnen wir wieder mit einer expliziten Betrachtung des Levels **DA** und definieren anschließend ein allgemeines Schema für die inneren Level der Hierarchie.

### 4.2.1 Varietät DA

Wir betrachten das Verbotsmuster $M^{\rho}_{DA1}$, das in Abbildung 4.14 dargestellt ist, welches aus $M_{DA1}$ durch Umkehren der Kanten entsteht. Dieses Verbotsmuster soll **DA** für fin-syntaktische Monoide charakterisieren. Eingehende Kanten von $+/-$ beziehungsweise $-/+$ geben an, dass man genau eines der Bilder dieser Knoten durch einen Pfad mit gleicher Beschriftung aus einem Startzustand erreicht und das Andere nicht, bei Abbildung des Musters in einen CMA. Analog gibt eine ausgehende mit $q$ beschriftete Kante an, dass man vom Bild dieses Knotens einen Endzustand unendlich oft erreicht.

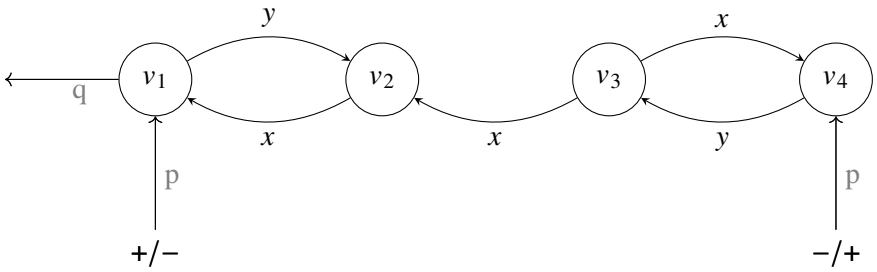



**Abbildung 4.14:** Verbotsmuster $M_{\mathbf{DA1}^\rho}$ für **DA**.

**Lemma 4.7** *Sei $\mathcal{A}$ ein CMA. Dann ist das reverse Verbotsmuster $M_{\mathbf{DA1}^\rho}$ in $\mathcal{A}$ genau dann enthalten, wenn $Synt_{fin}(L(\mathcal{A})) \notin \mathbf{DA}$ gilt.*

*Beweis.* Für die erste Richtung sei das reverse Verbotsmuster $M_{\mathbf{DA1}^\rho}$ in $\mathcal{A} = (P, \Sigma, \cdot, I, F)$ enthalten. Dann existieren Abbildungen $h : V \to Z$ und $g : X \to Synt_{fin}(L(\mathcal{A}))$, sodass die entsprechenden Bedingungen erfüllt sind. Dadurch folgt $h(v_s) \cdot^\rho g(x) \cdot^\rho g((xy)^\omega) \cdot^\rho g(y) = h(v_l)$, sowie $h(v_s) \cdot^\rho g(x) \cdot^\rho g((xy)^\omega x(xy)^\omega) \cdot^\rho g(y) = h(v_r)$. Weiter folgt:

$$p \cdot g(y) \cdot g((xy)^\omega) \cdot g(x) \cdot q \in L(\mathcal{A}) \iff p \cdot g(y) \cdot g((xy)^\omega x(xy)^\omega) \cdot g(x) \cdot q \notin L(\mathcal{A})$$

Also sind $g((xy)^\omega)$ und $g((xy)^\omega x(xy)^\omega)$ unterschiedliche Element in $Synt_{fin}(L(\mathcal{A}))$. Es wurde also eine Variablenbelegung gefunden, welche die $\omega$-Gleichung $[\![(xy)^\omega = (xy)^\omega x(xy)^\omega]\!]$ nicht erfüllt. Dies führt zu $Synt_{fin}(L(\mathcal{A})) \notin \mathbf{DA}$.

Für die andere Richtung sei $Synt_{fin}(L(\mathcal{A})) \notin \mathbf{DA}$. Dann existieren Elemente $\tilde{x}, \tilde{y} \in Synt_{fin}(L(\mathcal{A}))$, für die $(\tilde{x}\tilde{y})^\omega \neq (\tilde{x}\tilde{y})^\omega \tilde{x} (\tilde{x}\tilde{y})^\omega$ gilt. Dadurch existieren $p, q, r \in \Sigma^*$ mit:

$$p(\tilde{x}\tilde{y})^\omega rq^\omega \in L(\mathcal{A}) \iff p(\tilde{x}\tilde{y})^\omega \tilde{x}(\tilde{x}\tilde{y})^\omega rq^\omega \notin L(\mathcal{A})$$

In $\mathcal{A}$ existieren Zustände $z_0, z_0'$ mit $|\{z_0, z_0'\} \cap I| = 1$ und $z_0 \cdot p = z_0' \cdot p \cdot (xy)^\omega x$ und von diesem Zustand kann mit $rq^\omega$ ein Endzustand unendlich oft erreicht werden. Da $\omega$ der Idempotenz-Exponent ist und aufgrund des Co-Determinismus von CMAs folgt weiter $z_0 \cdot p \cdot (\tilde{x}\tilde{y})^\omega = z_0 \cdot p$, sowie $z_0' \cdot p \cdot (\tilde{y}\tilde{x})^\omega = z_0' \cdot p$. Abbildung 4.15 skizziert den relevanten Teil von $\mathcal{A}$.

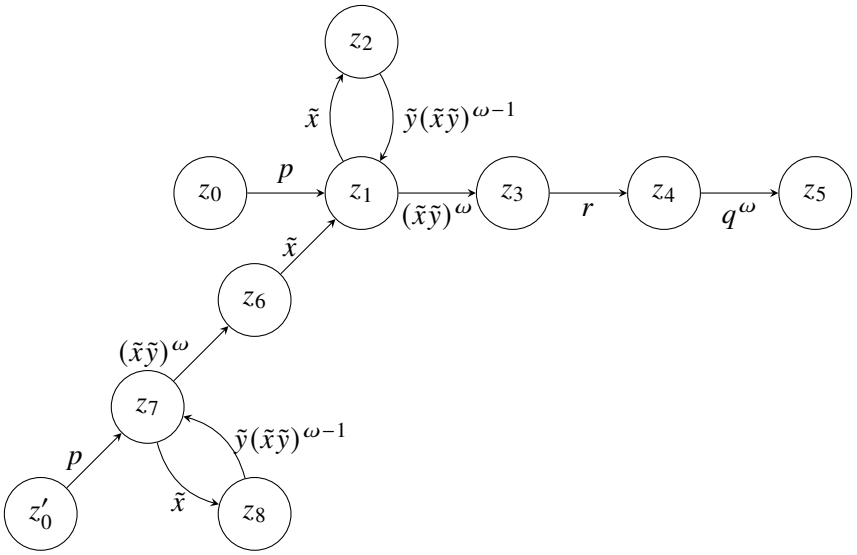



**Abbildung 4.15:** Teil des CMA $\mathcal{A}$, dessen fin-syntaktisches Monoid in **DA** ist.

Die folgenden Abbildungen beweisen nun das Enthaltensein von $M_{\mathbf{DA1}^\rho}$ in $\mathcal{A}$:

$$\begin{aligned} h(v_1) &= z_2 & g(x) &= (\tilde{x}\tilde{y})^\omega \tilde{x} \\ h(v_2) &= z_1 & g(y) &= (\tilde{y}\tilde{x})^{\omega-1} \tilde{y} \\ h(v_3) &= z_7 \\ h(v_4) &= z_8 \end{aligned}$$

□

Lemma 4.6 zeigt, dass das Verhalten der inf-syntaktischen Monoide für alle Level der Trotter-Weil-Hierarchie auf **DA** zurückgeführt werden kann. Dadurch kann ein Verbotsmuster für **DA** auch als Entscheidungskriterium für das Enthaltensein von inf-syntaktischen Monoiden in den anderen Leveln der Hierarchie benutzt werden. Im Rahmen dieser Arbeit ist es nicht gelungen, ein entsprechendes Verbotsmuster mit Korrektheitsbeweis zu erstellen. Als Entscheidungskriterium für die Mitgliedschaft eines inf-syntaktischen Monoids in **DA**, muss auf das entsprechend Teilwortmuster $\mathcal{P}_{\mathbf{DA}-inf}$ aus der Arbeit von Henriksson und Kufleitner zurückgegriffen werden [10].

### 4.2.2 Innere Varietäten der Trotter-Weil-Hierarchie

Für die vollständige Charakterisierung der Hierarchie fehlen nun noch die Verbotsmuster für die fin-syntaktischen Monoide der inneren Level. Diese können durch Umkehren der Verbotsmuster für die syntaktischen Monoide bei Sprachen über endlichen Wörtern erstellt werden, analog zum Resultat von Henriksson und Kufleitner für Teilwortmuster [10].

**Theorem 4.8** *Sei* $\mathbf{V}$ *eine Varietät mit* $\mathbf{J}_+ \subseteq \mathbf{V} \subseteq \mathbf{DA}$ *und* $\mathcal{A} = (Z, \Sigma, \cdot, I, F)$ *ein CMA. Sei weiter* $M = (V, E, v_s, v_l, v_r)$ *das Verbotsmuster für* $\mathbf{V}$ *bei DEAs. Dann ist* $M^\rho$ *in* $\mathcal{A}$ *genau dann enthalten, wenn* $Synt_{fin}(L(\mathcal{A})) \notin \mathbf{V}$ *gilt.*

*Beweis.* Zuerst ist anzumerken, dass $M^\rho$ die Varietät $\mathbf{V}$ für reverse DEAs charakterisiert, da reverse DEAs genau die umgekehrten Übergänge von DEAs enthalten und reverse Verbotsmuster die umgekehrten Kanten von Verbotsmustern. Für jeden Zustand $z \in Z$ definieren wir einen reverse DEA $\mathcal{A}_z = (Z, \Sigma, \cdot^\rho, I, \{z\})$. Das zugehörige syntaktische Monoid $Synt(L(\mathcal{A}_z))$ lässt sich auch darstellen durch:

$$\forall u, v \in \Sigma^* : \big(u \equiv_z v \iff (\forall x, y \in \Sigma^* : z \cdot^\rho xuy \in I \iff z \cdot^\rho xvy \in I)\big)$$

Nun ist $Synt(L(\mathcal{A}_z))$ ein Divisor von $Synt_{fin}(L(\mathcal{A}))$, denn für $z$ als Startzustand des finalen runs eines Wortes $ab^\omega$ und $x, y \in \Sigma^*$ gilt:

$$\begin{aligned} u \equiv_{fin} v &\implies \big(xuyab^\omega \in L(\mathcal{A}) \iff xvyab^\omega \in L(\mathcal{A})\big) \\ &\implies \big(z \cdot^\rho xuy \in I \iff z \cdot^\rho xvy \in I\big) \\ &\implies u \equiv_z v \end{aligned}$$

Die Existenz von $a, b \in \Sigma^*$ kann ohne Einschränkung gefordert werden. Außerdem ist $Synt_{fin}(L(\mathcal{A}))$ ein Divisor von $Synt(L(\mathcal{A}_{z_1})) \times \ldots \times Synt(L(\mathcal{A}_{z_n}))$ für $Z = \{z_1, \ldots, z_n\}$, denn falls $u \equiv_z v$ für alle $z \in Z$ gilt, folgt $xuyb^\omega \in L(\mathcal{A}) \iff xvyb^\omega \in L(\mathcal{A})$.

Für die erste Richtung des Beweises sei $M^\rho$ in $\mathcal{A}$ enthalten. Dann gilt, dass auf dem Pfad $h(v_s) \cdot q$ ein Endzustand $z_f \in F$ unendlich oft erreicht wird. Für dieses $z_f$ enthält auch der reverse DEA $\mathcal{A}_{z_f}$ das Muster $M^\rho$. Da $M^\rho$ für reverse DEAs die Varietät $\mathbf{V}$ charakterisiert, gilt damit $Synt(\mathcal{A}_{z_f}) \notin \mathbf{V}$. Nach Definition sind Varietäten unter Division abgeschlossen, weshalb auch $Synt_{inf}(L(\mathcal{A})) \notin \mathbf{V}$ gelten muss.

Für die zweite Richtung sei $Synt_{fin}(L(\mathcal{A})) \notin \mathbf{V}$. Dann können aufgrund der Abgeschlossenheiten von Varietäten unter endlichen direkten Produkten und Division nicht alle $Synt(L(\mathcal{A}_{z_i}))$ in $\mathbf{V}$ sein. Es existiert also ein $z \in Z$ mit $Synt(L(\mathcal{A}_z)) \notin \mathbf{V}$. Da $M^\rho$ das definierende Muster für reverse DEAs für $\mathbf{V}$ ist, ist $M^\rho$ in $\mathcal{A}_z$ enthalten. Weiter ist $M^\rho$ auch in $\mathcal{A}$ enthalten, da $\mathcal{A}_z$ ein Teil von $\mathcal{A}$ ist. □

# 5 Prüfen auf feste Verbotsmuster

Die in Kapitel 4 entwickelten Verbotsmuster können nun benutzt werden, um für gegebene Automaten zu entscheiden, ob das zugehörige syntaktische Monoid in einer beliebigen festen Varietät der Trotter-Weil-Hierarchie liegt. Dies soll einen Komplexitätsvorteil bringen im Vergleich zur Berechnung des Monoids aus dem Automaten und anschließender Prüfung der Omega-Gleichung der Varietät für alle möglichen Variablenbelegungen.

Wie in [16] für Faktorverbotsmuster in minimalen DEAs und in [10] für Teilwortmuster in DEAs und CMAs, sollen hier Algorithmen mit nichtdeterministisch logarithmischem Platz (NL) angegeben werden, die DEAs und CMAs auf die in dieser Arbeit entwickelten Verbotsmuster untersuchen. Für DEAs definieren wir Algorithmus 5.1. Es gibt dabei für jedes Verbotsmuster einen eigenen Algorithmus, die alle gleich arbeiten. Der jeweilige Algorithmus erhält einen DEA als Eingabe. Zuerst werden die Knoten des Musters nichtdeterministisch auf Zustände des Automaten abgebildet, also das Bild der Abbildung $h : V \to Z$ geraten. Dies passiert in den Zeilen 1 bis 3. Anschließend wird überprüft, ob die drei Kriterien des Verbotsmuster-Enthaltens für diese Abbildung $h$ erfüllt werden können. Dabei wird das erste Kriterium in den Zeilen 4 bis 8 überprüft, indem ein Wort geraten wird, das einen Pfad vom Startzustand zu dem Zustand beschriftet, auf den $v_s$ abgebildet wird. Berechnungspfade, bei denen $v_s$ auf einen vom Startzustand unerreichbaren Zustand abgebildet wird, landen hierbei in einer Endlosschleife. Die Zeilen 9 bis 15 testen analog das zweite Kriterium durch Raten eines Worts, mit dem Pfade beschriftet sind, die genau einen der Zustände $h(v_l)$ und $h(v_r)$ in einen Endzustand überführen. Der verbleibende Teil des Algorithmus betrachtet das dritte Kriterium, es wird also nach einer geeigneten Abbildung $g : X \to Synt(L(\mathcal{A}))$ gesucht. Hierbei werden nacheinander für alle Kantenbeschriftungen des Verbotsmusters die Kanten gespeichert, die mit dieser beschriftet sind und eine Pfadbeschriftung geraten, deren Pfad den Zustand auf den der Startknoten der Kante abgebildet wird in den Zustand überführt, auf den der Endknoten abgebildet wird.

In dieser Arbeit wurden auch Verbotsmuster für CMAs erstellt. Algorithmus 5.2 prüft analog zu Algorithmus 5.1 das Enthaltensein von reverse Verbotsmustern in CMAs. Es ergibt sich die folgende Äquivalenz.

**Lemma 5.1** *Das Verbotsmuster einer Varietät der Trotter-Weil-Hierarchie ist in einem DEA (beziehungsweise CMA) genau dann enthalten, wenn Algorithmus 5.1 (beziehungsweise Algorithmus 5.2) auf Eingabe dieses Automaten terminiert.*

*Beweis.* Wir beweisen die Aussage für DEAs. Der Beweis für CMAs funktioniert analog. Wenn ein Verbotsmuster $M$ in einem DEA enthalten ist, gibt es Abbildungen $\tilde{h}$ und $\tilde{g}$, die die entsprechenden Bedingungen erfüllen. In den Zeilen 1 bis 3 werden alle Möglichkeiten für $h$ geraten, also auch $\tilde{h}$. Für diese Funktion, muss auch gelten, dass $\tilde{h}(v_s)$ von einem Startzustand erreichbar ist und ein Wort existiert, das Pfade beschriftet, die genau von einem der Zustände $\tilde{h}(v_l)$ und $\tilde{h}(v_r)$ in einen Endzustand führen. Diese Worte werden in den folgenden Zeilen des Algorithmus also auch

**Algorithmus 5.1** NL-Algorithmus der einen gegebenen DEA auf das Enthalten des Verbotsmusters $M = (V, E, v_s, v_l, v_r)$ mit Beschriftungsvariablenmenge $X$ prüft.

**Input:** DEA $\mathcal{A} = (Z, \Sigma, \cdot, z_0, F)$

1: **for** $i = 1$ to $|V|$ **do**
2: | rate $z \in Z$
3: | $h_{v_i} := z$ // die Knoten in $V$ seien mit $v_1$ bis $v_{|V|}$ benannt
4: $s := z_0$
5: **repeat**
6: | rate $\sigma \in \Sigma$
7: | $s := s \cdot \sigma$
8: **until** $s = h_{v_s}$
9: $s_l := h_{v_l}$
10: $s_r := h_{v_r}$
11: **repeat**
12: | rate $\sigma \in \Sigma$
13: | $s_l := s_l \cdot \sigma$
14: | $s_r := s_r \cdot \sigma$
15: **until** $\big(s_l \in F \wedge s_r \in Z \setminus F\big) \vee \big(s_l \in Z \setminus F \wedge z_r \in F\big)$
16: **for all** Beschriftungen $x \in X$ **do**
17: | $j := 1$
18: | **for all** Kanten $(v, v') \in E$ **do** // hier gilt $v, v' \in \{v_1, \ldots, v_{|V|}\}$
19: | | **if** $b(v, v') = x$ **then**
20: | | | $s_{j,1} := h_v$
21: | | | $s_{j,2} := h_{v'}$
22: | | | $j := j + 1$
23: | **repeat**
24: | | rate $\sigma \in \Sigma$
25: | | **for** $k = 1$ to $j$ **do**
26: | | | $s_{k,1} := s_{k,1} \cdot \sigma$
27: | **until** $\forall m \in \{1, \ldots, j\} : s_{m,1} = s_{m,2}$

geraten. Da $\tilde{g}$ die dritte Bedingung des Verbotsmusterenthaltens erfüllt, gibt es für jedes $x \in X$, das im Bild $b(E)$ liegt, ein Element $s \in Synt(L(\mathcal{A}))$, das Pfade im Automaten beschriftet, die die Bilder der Startknoten der mit $x$ beschrifteten Kanten in die Bilder der Endknoten überführen. Der Berechnungspfad, der dieses Wort rät, braucht also $|s|$ Iterationen der Schleife in Zeile 23, bis die Schleifenbedingung erfüllt ist. Damit terminiert der Algorithmus nach endlich vielen Iterationen für endlich viele Beschriftungen aus $X$.

Wenn andererseits keine Funktionen existieren, die die entsprechenden Bedingung erfüllen, ist für jede Funktion $\tilde{h} : V \to Z$ mindestens eine der drei Bedingungen nicht erfüllbar. Sei $\tilde{h}$ eine beliebige Funktion von $V$ nach $Z$. Falls sie das erste Kriterium nicht erfüllt, gibt es kein Wort, das einen Pfad von $z_0$ zu $\tilde{h}(v_s)$ beschriftet und damit erfüllt kein geratenes Wort die Schleifenbedingung in Zeile 8 des Algorithmus. Diese Schleife wird also in allen Berechnungspfaden unendlich oft durchlaufen und der Algorithmus terminiert nicht. Falls $\tilde{h}$ das erste Kriterium erfüllt, das zweite jedoch nicht, wird in der ersten Schleife ein passendes Wort geraten und der Algorithmus erreicht die zweite Schleife. Da kein Wort existiert, welches das zweite Kriterium erfüllt, kann kein Wort geraten werden, mit dem die Schleifenbedingung in Zeile 15 wahr wird. Der Algorithmus befindet

**Algorithmus 5.2** NL-Algorithmus der einen gegebenen CMA auf das Enthalten des reverse Verbotsmusters $M^\rho = (V, E, v_s, v_l, v_r)$ mit Beschriftungsvariablenmenge $X$ prüft.

**Input:** CMA $\mathcal{A} = (Z, \Sigma, \cdot, I, F)$

1: **for** $i = 1$ to $|V|$ **do**
2: | rate $z \in Z$
3: | $h_{v_i} := z$ // die Knoten in $V$ seien mit $v_1$ bis $v_{|V|}$ benannt
4: $s := h_{v_s}$
5: **repeat**
6: | rate $\sigma \in \Sigma$
7: | $s := s \cdot \sigma$
8: **until** $s \in I$
9: $s_l := h_{v_l}$
10: $s_r := h_{v_r}$
11: **repeat**
12: | rate $\sigma \in \Sigma$
13: | $s_l := s_l \cdot \sigma$
14: | $s_r := s_r \cdot \sigma$
15: **until** $\big(s_l \in F \wedge s_r \in Z \setminus F\big) \vee \big(s_l \in Z \setminus F \wedge z_r \in F\big)$
16: **for all** Beschriftungen $x \in X$ **do**
17: | $j := 1$
18: | **for all** Kanten $(v, v') \in E$ **do** // hier gilt $v, v' \in \{v_1, \ldots, v_{|V|}\}$
19: | | **if** $b(v, v') = x$ **then**
20: | | | $s_{j,1} := h_v$
21: | | | $s_{j,2} := h_{v'}$
22: | | | $j := j + 1$
23: | **repeat**
24: | | rate $\sigma \in \Sigma$
25: | | **for** $k = 1$ to $j$ **do**
26: | | | $s_{k,1} := s_{k,1} \cdot \sigma$
27: | **until** $\forall m \in \{1, \ldots, j\} : s_{m,1} = s_{m,2}$

sich hier also in jedem Berechnungspfad in einer Endlosschleife. Falls die ersten beiden Kriterien durch $\tilde{h}$ erfüllt werden, das letzte jedoch nicht, gibt es mindestens eine Beschriftung $\tilde{x} \in X$, für die keine mögliche Funktion $\tilde{g} : X \to Synt(L(\mathcal{A}))$ alle entsprechenden Zustände ineinander überführt. Für die Iteration der Schleife in Zeile 16, bei der $x = \tilde{x}$ gilt, kann also kein Wort geraten werden, dass die Schleifenbedingung in Zeile 27 erfüllt. Der Algorithmus befindet sich demnach in einer Endlosschleife. □

**Theorem 5.2** *Das Problem, das syntaktisches Monoid eines gegeben DEA oder CMA auf Mitgliedschaft in einer gegebenen Varietät der Trotter-Weil-Hierarchie zu prüfen, liegt in NL.*

*Beweis.* Sei $Synt(L(\mathcal{A}))$ das syntaktische Monoid eines DEA $\mathcal{A}$. In Kapitel 4 wurde gezeigt, dass $Synt(L(\mathcal{A}))$ genau dann in einer Varietät der Trotter-Weil-Hierarchie liegt, wenn $\mathcal{A}$ das zugehörige Verbotsmuster nicht enthält. Lemma 5.1 zeigt, dass das Enthaltensein eines gegebenen Verbotsmusters einer Varietät der Trotter-Weil-Hierarchie in $\mathcal{A}$ durch Terminierung von Algorithmus 5.1 entschieden werden kann. Wir zeigen nun, dass Algorithmus 5.1 eine logarithmische Platzbeschränkung einhält. Es ist zu sehen, dass Algorithmus 5.1 in den Zeilen 1 bis 3 für jeden

Knoten des Verbotsmusters einen Zustand speichert. Diese werden für die gesamte Laufzeit des Algorithmus benötigt. Anschließend werden drei Zustände in den Variablen $s, s_l, s_r$ gespeichert. Die Speicherplätze hierfür können ab Zeile 16 wieder überschrieben werden. In der Schleife von Zeile 16 bis 27 werden für jede Kante mit einer bestimmten Kantenbeschriftung zwei Zustände gespeichert. Diese Speicherplätze können in der nächsten Iteration der Schleife wiederverwendet werden, da die Kantenbeschriftungen unabhängig voneinander sind. Damit werden höchstens $|V| + 2|E|$ Speicherplätze für Zustände benötigt und bis zu drei Speicherplätze für die Laufvariablen der Schleifen und die geratenen Buchstaben. Die Zustände sind Teil der Eingabe, deshalb können sie in logarithmischem Platz gespeichert werden. Der Platzbedarf ist also logarithmisch in der Eingabegröße. Es gilt also, dass das Problem des Enthaltenseins von Verbotsmustern in DEAs in NL liegt. Immerman und Szelepcsnyi zeigten, dass nichtdeterministische Platzklassen unter Komplement abgeschlossen sind [11], [22]. Das bedeutet insbesondere, dass NL=co-NL gilt. Damit liegt auch das Problem des Nicht-Enthaltenseins von Verbotsmustern in DEAs in NL. Schließlich folgt, dass das Problem der Zugehörigkeit von $Synt(L(\mathcal{A}))$ zu einer Varietät der Trotter-Weil-Hierarchie in NL liegt.

Sei nun $Synt(L(\mathcal{A}))$ das syntaktische Monoid eines CMA $\mathcal{A}$. Lemma 4.6 zeigt, dass $Synt(L(\mathcal{A}))$ genau dann in einer Varietät der Trotter-Weil-Hierarchie enthalten ist, wenn $Synt_{fin}(L(\mathcal{A}))$ in dieser Varietät und $Synt_{inf}(L(\mathcal{A}))$ in **DA** enthalten ist. Analog zum Fall für DEAs, kann Algorithmus 5.2 in logarithmischem Platz das reverse Verbotsmuster der Varietät in einem CMA finden und es kann gleichermaßen gezeigt werden, dass das Problem der Mitgliedschaft von $Synt_{fin}(L(\mathcal{A}))$ in einer Varietät der Trotter-Weil-Hierarchie in NL liegt. Proposition 49 in [10] zeigt, dass das Überprüfen eines CMA auf das Teilwortmuster $\mathcal{P}_{\mathbf{DA}-inf}$ in NL liegt und damit auch das Überprüfen auf Mitgliedschaft von $Synt_{inf}(L(\mathcal{A}))$ in **DA**. NL ist unter Komposition abgeschlossen, deshalb kann man mit der sequenziellen Überprüfung der Mitgliedschaft von $Synt_{fin}(L(\mathcal{A}))$ in einer Varietät der Trotter-Weil-Hierarchie und der Mitgliedschaft von $Synt_{inf}(L(\mathcal{A}))$ in **DA**, auch die Mitgliedschaft von $Synt(L(\mathcal{A}))$ in der Varietät nichtdeterministisch mit logarithmisch beschränktem Platz durchführen. □

# 6 Vergleich verschiedener Verbotsmusterarten

Neben den in Kapitel 4 vorgestellten Verbotsmustern, existieren auch Verbotsmuster, die Teilwortbeziehungen zwischen den Kantenbeschriftungen fordern. Henriksson und Kufleitner definieren solche Muster mit Teilwortbeziehungen für die Trotter-Weil-Hierarchie [10]. Allgemeine Teilwort-Verbotsmuster sind dort folgendermaßen definiert.

**Definition 6.1 (Verbotsmuster mit Teilwortbeziehungen [10])** *Sei $X$ eine Menge mit einer partiellen Ordnung $\preceq$. Ein Typ 2 Teilwort-Verbotsmuster $\mathcal{P} = (\mathcal{S}, j \not\preceq k)$ besteht aus einem endlichen partiellen Semi-DEA $\mathcal{S} = (V, X, \circ)$ und zwei Zuständen $j, k \in V$. $\mathcal{P}$ ist in einem DEA $\mathcal{A} = (Z, \Sigma, \cdot, z_0, F)$ genau dann enthalten, wenn ein Homomorphismus $g : X^* \to \Sigma^*$ existiert, sodass $x \preceq y$ impliziert, dass $g(x)$ ein Teilwort von $g(y)$ ist und ein Semi-DEA-Homomorphismus $h : S \to \mathcal{A}^h$ existiert, sodass $h(j) \not\leq_{\mathcal{A}} h(k)$ gilt und für alle $l \in V$ der Zustand $h(l)$ von $z_0$ aus erreichbar ist.*

In diesem Abschnitt wird ein Vergleich zwischen den beiden Musterarten bei den Verbotsmustern für die Trotter-Weil-Hierarchie in Bezug auf Größe, Komplexität der zugehörigen Algorithmen, sowie Größe der Zeugen angeführt. Zur besseren Unterscheidung werden wir die Muster aus dieser Arbeit, die keine Teilwortbeziehungen fordern, Faktormuster nennen. Da sich die Schnittlevel auch anhand der Muster der Ecklevel charakterisieren lassen, betrachten wir stellvertretend explizit nur die Muster für die Level $R_m$ und $L_m$ mit $m \geq 2$.

## 6.1 Größenvergleich

Zuerst betrachten wir die Größen der entwickelten Verbotsmuster. Hierbei sind die drei Kriterien Anzahl der Variablen, Anzahl der Knoten und Anzahl der Kanten interessant.

| Level | Musterart | Variablenzahl | Knotenzahl | Kantenzahl |
|---|---|---|---|---|
| $R_m$ | Teilwortmuster | $m$ | $m + \lceil \frac{m}{2} \rceil$ | $3m - 3 + (m \mod 2)$ |
| | Faktormuster | $2m - 2$ | $O(4^m)$ | $O(4^m)$ |
| $L_m$ | Teilwortmuster | $m$ | $m + \lfloor \frac{m}{2} \rfloor + 1$ | $3m - 1 - (m \mod 2)$ |
| | Faktormuster | $2m - 2$ | $O(4^m)$ | $O(4^m)$ |

**Tabelle 6.1:** Vergleich der Verbotsmusterarten in Bezug auf die Größe der Muster.

Für die Level $R_m$ und $L_m$ benötigen die Teilwortmuster die Variablenmenge $\{x, e_1, \ldots, e_{\lfloor m/2 \rfloor}, f_1, \ldots, f_{\lfloor (m-1)/2 \rfloor}\}$, beziehungsweise $\{x, e_1, \ldots, e_{\lfloor (m-1)/2 \rfloor}, f_1, \ldots, f_{\lfloor m/2 \rfloor}\}$, also $m$ viele Variablen. Die Faktormuster benötigen dagegen die $2m-2$ vielen Variablen der Variablenmengen $\{z, x_1, \ldots, x_{m-1}, y_1, \ldots, y_{m-2}\}$ beziehungsweise $\{z, x_1, \ldots, x_{m-1}, y_1, \ldots, y_{m-2}\}$. Die Variablenzahl ist also bei beiden Musterarten linear in $m$ und bei den Faktormustern ungefähr doppelt so groß wie bei den Teilwortmustern.

Bei der Anzahl der Knoten ist zu sehen, dass die Verbotsmuster mit Teilwortbeziehungen $l + 2l' + 2$ viele besitzen, wobei $l$ der höchste Index einer $e_i$-Variable und $l'$ der höchste Index einer $f_i$-Variable ist. Damit ergibt sich eine Knotenzahl zwischen $m + \frac{m}{2}$ und $m + \frac{m}{2} + 1$, abhängig von der Parität von $m$ und davon, ob es ein linkes oder rechtes Ecklevel ist. Das Faktormuster wird für ein Level $R_m$ aus einem Grundmuster mit $3m-3$ Knoten aufgebaut, bei dem Kanten durch weitere Muster ersetzt werden. Das Ersetzen einer Kante mit der Beschriftung $e_i$ führt zu $4i-6$ weiteren Knoten und das Ersetzen einer Kante $f_i$ führt zu $2i-3$ weiteren Knoten. Hier ist anzumerken, dass diese allgemeinen Formeln auch die Muster $M_{e_2}$ und $M_{f_2}$, die das Rekursionsende des Einsetzens darstellen, korrekt repräsentieren. Es ist noch zu ermitteln, wie oft jedes Kantenmuster eingesetzt wird. Seien $a_{e_i}$ und $a_{f_i}$ die Anzahlen, wie oft eine Kante mit der Beschriftung $e_i$ beziehungsweise $f_i$ in einem Muster ersetzt wird. Eine Kante mit der Beschriftung $e_i$ wird einmal im Grundmuster ersetzt, sowie zweimal in jedem eingesetzten Kantenmuster, das für eine $e_j$-Kante mit $j > i$ eingesetzt wurde, und einmal in jedem eingesetzten Kantenmuster, das für eine $f_j$-Kante mit $j > i$ eingesetzt wurde. Gleiches gilt für eine Kante mit der Beschriftung $f_i$, mit dem einzigen Unterschied, dass diese zweimal im Grundmuster ersetzt wird. Demnach gilt $a_{e_m} = 1$ und $a_{f_m} = 0$, sowie $a_{e_i} = 1 + \sum_{j=i+1}^{m} (2a_{e_j} + a_{f_j})$ und $a_{f_i} = 2 + \sum_{j=i+1}^{m} (2a_{e_j} + a_{f_j})$ für $i < m$. Als Formel für die Knotenzahl $Kn_{R_m}$ des Verbotsmusters für $R_m$ ergibt sich:

$$Kn_{R_m} = 3m - 3 + \sum_{i=2}^{m} \left( a_{e_i} \cdot (4i-6) + a_{f_i} \cdot (2i-3) \right)$$

Die Werte von $a_{e_i}$ und $a_{f_i}$ für $i < m$ sollen nun noch genauer analysiert werden. Für $i = m-1$ erhalten wir aus der Formel explizit die Werte $a_{e_{m-1}} = 3$ und $a_{f_{m-1}} = 4$. Für $i < m-1$ können wir damit nun rekursiv Werte berechnen. Offensichtlich gilt $a_{e_i} = a_{f_i} - 1$ für $i < m$ und $a_{e_i} = a_{f_i} + 1$ für $i = m$. Wir können also folgendermaßen umformen:

$$
\begin{aligned}
a_{e_i} &= 1 + \sum_{j=i+1}^{m} (2a_{e_j} + a_{f_j}) \\
&= 1 + \sum_{j=i+1}^{m} (3a_{e_i}) + \sum_{j=i+1}^{m-1} (1) - 1 \\
&= \sum_{j=i+1}^{m} (3a_{e_i}) + m - i - 1 \\
&= 3a_{e_{i+1}} + \sum_{j=i+2}^{m} (3a_{e_j}) + m - i - 1 \\
&= 3a_{e_{i+1}} + \left( \sum_{j=i+2}^{m} (3a_{e_j}) + m - i - 2 \right) + 1 \\
&= 3a_{e_{i+1}} + a_{e_{i+1}} + 1 \\
&= 4a_{e_{i+1}} + 1
\end{aligned}
$$

$$
\begin{aligned}
a_{f_i} &= 2 + \sum_{j=i+1}^{m} (2a_{e_j} + a_{f_j}) \\
&= 2 + \sum_{j=i+1}^{m} (3a_{f_j}) - \sum_{j=i+1}^{m-1} (2) + 2 \\
&= \sum_{j=i+1}^{m} (3a_{f_i}) - 2m + 2i + 6 \\
&= 3a_{f_{i+1}} + \sum_{j=i+2}^{m} (3a_{f_j}) - 2m + 2i + 6 \\
&= 3a_{f_{i+1}} + \left( \sum_{j=i+2}^{m} (3a_{f_j}) - 2m + 2i + 8 \right) - 2 \\
&= 3a_{f_{i+1}} + a_{f_{i+1}} - 2 \\
&= 4a_{f_{i+1}} - 2
\end{aligned}
$$

Für $f : \mathbb{N} \to \mathbb{N}$, $n \mapsto a_{e_{m-n}}$ und $g : \mathbb{N} \to \mathbb{N}$, $n \mapsto a_{f_{m-n}}$ können wir $f(n), g(n) \leq 4^n - n$ und damit $f(n), g(n) \in O(4^n)$ zeigen. Es folgt $Kn_{R_m} \in O(4^m)$. Für ein Level $L_m$ hat das Grundmuster $3m-2$ viele Knoten und für die Anzahl der Kantenersetzungen gilt $a_{e_m} = 0, a_{f_m} = 2$. In den anderen Aspekten stimmt die Berechnung mit der für $Kn_{R_m}$ überein. Damit ergibt sich analog $Kn_{L_m} \in O(4^m)$ für die Kantenzahl $Kn_{L_m}$ des Verbotsmusters des Levels $L_m$. Die Faktorverbotsmuster sind mit einer in $m$ exponentiellen Knotenzahl also deutlich größer als die Teilwortmuster, die nur linear vieler Knoten bedürfen.

Ähnlich wie bei der Knotenzahl, kann man die Kantenzahl der Teilwortmuster aus der Anzahl der Variablen berechnen. Es sei wieder $l$ der höchste Index einer $e_i$-Variablen und $l'$ der höchste Index einer $f_i$-Variablen. Dann hat das Teilwortmuster für ein Level $R_m$ oder $L_m$ die Kantenzahl

$2l + 4l' + 1$. Im Fall von $R_m$ ergibt sich daraus $3m - 3$ für gerade $m$ und $3m - 2$ für ungerade $m$. Im Fall von $L_m$ ergibt sich daraus $3m - 1$ für gerade $m$ und $3m - 2$ für ungerade $m$. Für die Kantenzahlermittlung der rekursiven Muster, bemerken wir zuerst, dass nur diejenigen Kanten im finalen Verbotsmuster enthalten sind, die mit $x_i$, $y_i$ oder $z$ beschriftet sind, wohingegen alle Kanten mit einer $e_i$- oder $f_i$-Beschriftung während des rekursiven Aufbaus ersetzt werden. Im Grundmuster für $R_m$ und $L_m$ existiert nur je eine bleibende Kante. Kantenersetzungen für Kanten mit der Beschriftung $e_i$ bringen vier weitere und Kantenersetzungen für Kanten mit der Beschriftung $f_i$ drei weitere bleibende Kanten. Die Anzahlen der Kantenersetzungen $a_{e_i}$ und $a_{f_i}$ mit $i < m$, sowie $a_{e_m}$ und $a_{f_m}$ stimmen mit denen aus der Berechnung der Knotenzahl überein. Die Kantenzahlen $Ka_{R_m}$ und $Ka_{L_m}$ belaufen sich auf

$$Ka_{R_m} = 1 + \sum_{i=2}^{m} (4 \cdot a_{e_i} + 3 \cdot a_{f_i})$$

$$Ka_{L_m} = 1 + \sum_{i=2}^{m} (4 \cdot a_{e_i} + 3 \cdot a_{f_i})$$

mit den entsprechenden Werten für $a_{e_i}$ und $a_{f_i}$, sowie $a_{e_m}$ und $a_{f_m}$. Es gilt $Ka_{R_m}, Ka_{L_m} \in O(4^m)$ und damit haben die Faktormuster eine exponentielle Kantenzahl gegenüber der linearen Kantenzahl der Teilwortmuster.

## 6.2 Algorithmenvergleich

In Kapitel 5 sind Algorithmen angegeben, die Automaten auf das Enthalten von Faktorverbotsmustern prüfen. Die Arbeit von Henriksson und Kufleitner enthält einen entsprechenden Algorithmus für das Überprüfen auf Teilwortmuster [10]. Hier wird ein Vergleich der Komplexität der beiden Algorithmen angeführt. Die Algorithmen benötigen alle logarithmisch viel Platz in der Eingabegröße. Im Gegensatz zu den Algorithmen für Teilwortmuster, konnte bei den Algorithmen für die Faktormuster die Optimierung eingebaut werden, gleichzeitig nur diejenigen Kanten zu speichern, die gleich beschriftet sind und den Speicherplatz anschließend für die Kanten mit der nächsten Beschriftung wiederzuverwenden. Für die Platzbedarfsschranke macht dies jedoch keinen Unterschied. Es wird jeweils eine konstante Menge an Zuständen gespeichert und damit sind die Platzbedarfe in $O(\log |Z|)$.

Von Interesse sind jedoch insbesondere deterministische Komplexitätsangaben. Es ist bekannt, dass NL eine Teilmenge der Klasse deterministisch polynomielle Zeit (P) ist und deshalb müssen deterministische Polynomialzeitalgorithmen existieren, die Automaten auf feste Verbotsmuster prüfen. Diese Algorithmen sollen nun definiert werden. Um die Verständlichkeit der Algorithmen zu verbessern, definieren wir für gegebene Automaten und Verbotsmuster $H := \{h_0, \ldots, h_d\}$ als die Menge aller Funktionen von $V$ nach $Z$ und benennen die Knoten in $V$ mit $v_0$ bis $v_e$, wobei $v_0 := v_s$, $v_1 := v_l$ und $v_2 := v_r$ gelten soll (im Fall von Teilwortmustern soll stattdessen $v_0 := v_j$ und $v_1 := v_k$ gelten). Damit zeigen nun Algorithmus 6.1 und Algorithmus 6.2 deterministische Algorithmen, deren Laufzeit polynomiell in der Eingabegröße ist, die das Gleiche berechnen wie Algorithmus 5.1 beziehungsweise Algorithmus 1 in [10], also DEAs auf das Enthalten von Faktor- oder TeilwortVerbotsmustern prüfen. Analoge Polynomialzeitalgorithmen können auch für das Prüfen von CMAs auf reverse Verbotsmuster erstellt werden.

**Algorithmus 6.1** P-Algorithmus der einen gegebenen DEA auf das Enthalten des Verbotsmusters $M = (V, E, v_s, v_l, v_r)$ mit Beschriftungsvariablenmenge $X$ prüft.

**Input:** DEA $\mathcal{A} = (Z, \Sigma, \cdot, z_0, F)$

1: $M := \big(h_i(v_j)\big)_{i\in\{0,\dots,d\}, j\in\{0,\dots,e\}}$ // Matrix mit den Werten für jede mögliche Funktion $\tilde{h} \in H$
2: $S := \big(\{(z_0, \varepsilon)\}\big)_{i\in\{0,\dots,d\}}$; $S' := \big(\{\}\big)_{i\in\{0,\dots,d\}}$; $S'' := S$ // Spaltenvektoren
3: $S_l := \big(\{(M[i][1], \varepsilon)\}\big)_{i\in\{0,\dots,d\}}$; $S'_l := \big(\{\}\big)_{i\in\{0,\dots,d\}}$; $S''_l := S_l$
4: $S_r := \big(\{(M[i][2], \varepsilon)\}\big)_{i\in\{0,\dots,d\}}$; $S'_r := \big(\{\}\big)_{i\in\{0,\dots,d\}}$; $S''_r := S_r$
5: **for all** Beschriftungen $x \in X$ **do**
6: $\quad c_x := 0$
7: **for all** Kanten $(v_t, v_u) \in E$ **do** // hier gilt $t, u \in \{0, \dots, e\}$
8: $\quad x := b(v_t, v_u)$
9: $\quad c_x := c_x + 1$
10: $\quad S_{x,c_x} := \big(\{(M[i][t], M[i][u], \varepsilon)\}\big)_{i\in\{0,\dots,d\}}$
11: $\quad S'_{x,c_x} := \big(\{\}\big)_{i\in\{0,\dots,d\}}$
12: $\quad S''_{x,c_x} := S$
13: **repeat**
14: $\quad$ **for all** Buchstaben $\sigma \in \Sigma$ **do**
15: $\quad\quad$ **for all** Zeilen $i \in \{0, \dots, d\}$ **do**
16: $\quad\quad\quad$ **for all** Zustand-Wort-Tupel $(z, w) \in S[i]$ **do** // $S[i]$ ist die $i$-te Zeile von $S$
17: $\quad\quad\quad\quad S'[i] := S'[i] \cup \{(z \cdot \sigma, w\sigma)\}$ // $w$ und $\sigma$ werden konkateniert
18: $\quad\quad\quad$ **for all** Zustand-Wort-Tupel $(z, w) \in S_l[i]$ **do**
19: $\quad\quad\quad\quad S'_l[i] := S'_l[i] \cup \{(z \cdot s, w\sigma)\}$
20: $\quad\quad\quad$ **for all** Zustand-Wort-Tupel $(z, w) \in S_r[i]$ **do**
21: $\quad\quad\quad\quad S'_r[i] := S'_r[i] \cup \{(z \cdot s, w\sigma)\}$
22: $\quad\quad\quad$ **for all** Beschriftungen $x \in X$ **do**
23: $\quad\quad\quad\quad$ **for** $k = 0$ to $c_x$ **do**
24: $\quad\quad\quad\quad\quad$ **for all** Zustand-Zustand-Wort-Tupel $(z, z', w) \in S_{x,k}[i]$ **do**
25: $\quad\quad\quad\quad\quad\quad S'_{x,k}[i] := S'_{x,k}[i] \cup \{(z \cdot \sigma, z', w\sigma)\}$
26: $\quad S := S'$; $S' := \{\}$
27: $\quad S_l := S'_l$; $S'_l := \{\}$
28: $\quad S_r := S'_r$; $S'_r := \{\}$
29: $\quad$ **for all** Zeilen $i \in \{0, \dots, d\}$ **do**
30: $\quad\quad S''[i] := S''[i] \cup S[i]$
31: $\quad\quad S''_l[i] := S''_l[i] \cup S_l[i]$
32: $\quad\quad S''_r[i] := S''_r[i] \cup S_r[i]$
33: $\quad$ **for all** Beschriftungen $x \in X$ **do**
34: $\quad\quad$ **for** $k = 0$ to $c_x$ **do**
35: $\quad\quad\quad S_{x,k} := S'_{x,k}$
36: $\quad\quad\quad S'_{x,k} := \{\}$
37: $\quad\quad\quad$ **for all** Zeilen $i \in \{0, \dots, d\}$ **do**
38: $\quad\quad\quad\quad S''_{x,k}[i] := S''_{x,k}[i] \cup S_{x,k}[i]$
39: **until** $\exists j \in \{0, \dots, d\} : \Big(\big(\exists w' \in \Sigma^* : (M[j][0], w') \in S''[j]\big) \wedge \big(\exists w' \in \Sigma^* \exists z_1, z_2 \in Z : (z_1, w') \in S''_l[j] \wedge (z_2, w') \in S''_r[j] \wedge |\{z_1, z_2\} \cap F| = 1\big) \wedge \big(\forall y \in X : \exists w' \in \Sigma^* \forall m \in \{0, \dots, c_y\} : (M[j][u], M[j][u], w') \in S''_{y,m}[j]\big)\Big)$

**Algorithmus 6.2** P-Algorithmus der einen gegebenen DEA auf das Enthalten des Teilwort-Verbotsmusters $\mathcal{P} = (\mathcal{S}, j \not\leq k)$ mit $\mathcal{S} = (V, X, \circ)$ prüft.

**Input:** DEA $\mathcal{A} = (Z, \Sigma, \cdot, z_0, F)$

$M := \left(h_i(v_j)\right)_{i\in\{0,\dots,d\}, j\in\{0,\dots,e\}}$
**for** $a = 0$ to $e$ **do**
  $S_a := \left(\{(z_0, \varepsilon)\}\right)_{i\in\{0,\dots,d\}}$; $S'_a := \left(\{\}\right)_{i\in\{0,\dots,d\}}$; $S''_a := S_a$
$S_j := \left(\{(M[i][1], \varepsilon)\}\right)_{i\in\{0,\dots,d\}}$; $S'_j := \left(\{\}\right)_{i\in\{0,\dots,d\}}$; $S''_j := S_j$
$S_k := \left(\{(M[i][2], \varepsilon)\}\right)_{i\in\{0,\dots,d\}}$; $S'_k := \left(\{\}\right)_{i\in\{0,\dots,d\}}$; $S''_k := S_k$
**for all** Beschriftungen $x \in X$ **do**
  $c_x := 0$
**for all** Kanten $(v_t, x, v_u) \in \circ$ **do**
  $c_x := c_x + 1$
  $S_{x,c_x} := \left(\{(M[i][t], M[i][u], \varepsilon)\}\right)_{i\in\{0,\dots,q\}}$; $S'_{x,c_x} := \left(\{\}\right)_{i\in\{0,\dots,d\}}$; $S''_{x,c_x} := S$
**repeat**
  **for all** Buchstaben $\sigma \in \Sigma$ **do**
    **for all** Zeilen $i \in \{0, \dots, d\}$ **do**
      **for** $a = 0$ to $e$ **do**
        **for all** Zustand-Wort-Tupel $(z, w) \in S_a[i]$ **do**
          $S'_a[i] := S'_a[i] \cup \{(z \cdot \sigma, w\sigma)\}$
      **for all** Zustand-Wort-Tupel $(z, w) \in S_j[i]$ **do**
        $S'_j[i] := S'_j[i] \cup \{(z \cdot s, w\sigma)\}$
      **for all** Zustand-Wort-Tupel $(z, w) \in S_k[i]$ **do**
        $S'_k[i] := S'_k[i] \cup \{(z \cdot s, w\sigma)\}$
      **for all** Beschriftungen $x \in X$ **do**
        **for** $f = 0$ to $c_x$ **do**
          **for all** Zustand-Zustand-Wort-Tupel $(z, z', w) \in S_{x,f}[i]$ **do**
            $S'_{x,f}[i] := S'_{x,f}[i] \cup \{(z \cdot \sigma, z', w\sigma)\}$
  **for** $a = 0$ to $e$ **do**
    $S_a := S'_a$; $S'_a := \{\}$;
    **for all** Zeilen $i \in \{0, \dots, d\}$ **do**
      $S''_a[i] := S''_a[i] \cup S_a[i]$
  $S_j := S'_j$; $S'_j := \{\}$; $S_k := S'_k$; $S'_k := \{\}$
  **for all** Zeilen $i \in \{0, \dots, d\}$ **do**
    $S''_j[i] := S''_j[i] \cup S_j[i]$
    $S''_k[i] := S''_k[i] \cup S_k[i]$
  **for all** Beschriftungen $x \in X$ **do**
    **for** $f = 0$ to $c_x$ **do**
      $S_{x,f} := S'_{x,f}$; $S'_{x,f} := \{\}$
      **for all** Zeilen $i \in \{0, \dots, d\}$ **do**
        $S''_{x,f}[i] := S''_{x,f}[i] \cup S_{x,f}[i]$

**until** $\exists c \in \{0, \dots, d\} : \Big( \big(\forall b \in \{0, \dots, e\}\, \exists w' \in \Sigma^* : (M[c][b], w') \in S''_b[c]\big) \wedge \big(\exists w' \in \Sigma^*\, \exists z_1 \in F\, \exists z_2 \in Z \setminus F : (z_1, w') \in S''_j[c] \wedge (z_2, w') \in S''_k[c]\big) \wedge \big(\forall y \in X\, \exists (s_{y_0}, s_{y_0}, w_y), \dots, (s_{y_{c_y}}, s_{y_{c_y}}, w_y) : \big(\forall 0 \leq i \leq c_y : (s_{y_i}, s_{y_i}, w_y) \in S_{y,i}[c]''\big) \wedge \big(\forall z \in X (y \preceq z \implies \exists u, v \in \Sigma^* : w_y = u w_z v)\big)\big)\Big)$

An den Stellen, an denen die nichtdeterministischen Algorithmen einen Wert raten, werden bei den hier aufgeführten Algorithmen stattdessen über eine Schleife alle möglichen Werte nacheinander gewählt und gespeichert. Da es sich jeweils um endlich viele mögliche Werte handelt, ist dies bekanntlich in polynomieller Zeit möglich und die Algorithmen terminieren auf gleicher Eingabe genau dann, wenn Algorithmus 5.1 beziehungsweise Algorithmus 1 aus [10] terminieren. Im Gegensatz zu den nichtdeterministischen Algorithmen, können die drei Bedingungen des Verbotsmuster-Enthaltens nicht nacheinander geprüft werden, da der Algorithmus sonst in Endlosschleifen landen könnte, bevor die richtige Lösung gefunden ist. Es müssen also alle drei Kriterien gleichzeitig geprüft werden und somit alle möglichen Werte dafür gespeichert werden. Dies führt dazu, dass für einige Kriterien weitere Kandidaten berechnet werden, obwohl bereits eine Lösung für diese gefunden ist. Des Weiteren kann somit auch kein Optimierungsvorteil der Verbotsmuster ohne Teilwortbeziehungen gewonnen werden, da alle Kantenabbildungen gleichzeitig gespeichert und berechnet werden müssen. Die beiden Algorithmen, Algorithmus 6.1 und Algorithmus 6.2, unterscheiden sich auch darin, dass in Algorithmus 6.2 die Erreichbarkeit vom Startzustand für alle Knoten geprüft werden muss, statt nur für einen, was bedeutet, dass hier $|V|$ viele Wörter gefunden werden müssen statt einem. Außerdem müssen in Algorithmus 6.2 die geforderten Teilwortbeziehungen überprüft werden, was einen zusätzlichen Faktor $|X|$ in der Iterationsbedingungsprüfung bewirkt. Die Laufzeit der Algorithmen soll nun noch genauer analysiert und verglichen werden.

Wir betrachten Algorithmus 6.1. In den Zeilen 1 bis 12 werden die benötigten Variablen initialisiert. Die Matrix $M$ enthält alle möglichen Abbildungen von $V$ nach $Z$. Dabei definiert jede Zeile der Matrix eine Abbildung, indem in jeder Zelle das Bild eines Knotens unter dieser Abbildung gespeichert ist. Um die drei Kriterien für das Enthalten eines Verbotsmusters zu überprüfen, werden Spaltenvektoren gespeichert, die das Bild bestimmter Knoten unter jeder Abbildung enthalten. Diese Initialisierungsphase des Algorithmus läuft in $O(|H| \cdot |V| + |H| + |X| + |H| \cdot |E|)$. Die Schleife von Zeile 13 bis Zeile 39 iteriert so oft, bis alle drei Kriterien erfüllt sind. Im Fall, dass das gesuchte Verbotsmuster im eingegebenen Automaten enthalten ist, sind alle Kriterien nach $I := \max\left\{|p|, |q|, \max\{|g(x)| \mid x \in X\}\right\}$ Iterationen erfüllt. Im $n$-ten Schleifendurchlauf werden $O(|\Sigma| \cdot |H| \cdot |E| \cdot |\Sigma|^n + |H| \cdot |E|)$ Operationen ausgeführt. Die Schleife benötigt also $O(|\Sigma| \cdot |H| \cdot |E| \cdot \sum_{f=0}^{I} |\Sigma|^f + I \cdot |H| \cdot |E|)$ Zeitaufwand. Das Abbruchkriterium der Schleife in Zeile 39 wird $I$ mal geprüft. Die $n$-te überprüfung benötigt $O(|H| \cdot |E| \cdot \sum_{f=0}^{n} |\Sigma|^f)$ Operationen. Die gesamten Überprüfungen laufen damit in $O\big(|H| \cdot |E| \cdot \sum_{f=0}^{I} \big((I-f)|\Sigma|^f\big)\big)$. Das Verbotsmuster ist fest, wodurch $|H|$, $|V|$ und $|X|$ konstant sind. Die Laufzeit der Schleife ist die Dominierende und dadurch erhalten wir für die Gesamtlaufzeit des Algorithmus $O\big(|Z|^{|V|} \cdot \sum_{f=1}^{I+1} |\Sigma|^f\big) = O\big(|Z|^{|V|} \cdot |\Sigma|^{I+1}\big)$. Es handelt sich also um eine polynomielle Laufzeit in der Eingabegröße.

Nun betrachten wir Algorithmus 6.2. Die Initialisierungsphase in den Zeilen 1 bis 10 läuft in $O(|H| \cdot |V| + |V| \cdot |H| + |X| + |H| \cdot |\circ|)$. Wenn das Verbotsmuster in einem Automaten enthalten ist, gibt es Worte $p_0, \ldots, p_e, q \in \Sigma^*$, sodass $\forall 0 \leq i \leq e : z_0 \cdot p_i = h(v_i)$, sowie $h(v_j) \cdot q \in F$ und $h(v_k) \cdot q \in Z \setminus F$ gilt. Bis eine Lösung gefunden ist, durchläuft die Schleife in den Zeilen 11 bis 38 dann $J := \max\left\{|q|, \max\{|p_i| \mid 0 \leq i \leq e\}, \max\{|g(x)| \mid x \in X\}\right\}$ Iterationen. Im $n$-ten Schleifendurchlauf werden $O\big(|\Sigma| \cdot |H| \cdot |\circ| \cdot |\Sigma|^n (|V|+|E|) + |H| \cdot (|V|+|\circ|)\big)$ Operationen ausgeführt. Die Schleife benötigt also einen Zeitaufwand von $O\big(|\Sigma| \cdot |H| \cdot |\circ| \cdot \sum_{f=0}^{J} |\Sigma|^f + J \cdot |H| \cdot (|V|+|\circ|)\big)$. Das Abbruchkriterium in Zeile 38 wird $J$ mal geprüft, wobei $O(|H| \cdot (|\circ| + |V|) \cdot \sum_{f=0}^{n} |\Sigma|^f)$ viele Operationen beim $n$-ten Mal ausgeführt werden. Damit ergibt sich für die Überprüfung insgesamt eine

Laufzeit in $O(|H| \cdot (|\circ| + |V|) \cdot \sum_{f=0}^{J} \left((J-f)|\Sigma|^f\right))$. Als Gesamtlaufzeit des Algorithmus erhält man aus der domienierenden Laufzeit der Schleife entsprechend $O\left(|Z|^{|V|} \cdot \sum_{f=1}^{J+1} |\Sigma|^f\right) = O\left(|Z|^{|V|} \cdot |\Sigma|^{J+1}\right)$ nach Entfernen der Konstanten.

Vergleichend ergibt sich eine Laufzeit in $O\left(|Z|^{|V|} \cdot |\Sigma|^{I+1}\right)$ für das Finden eines Faktorverbotsmusters gegen eine Laufzeit in $O\left(|Z|^{|V|} \cdot |\Sigma|^{J+1}\right)$ für das Finden eines Teilwortmusters. Der Unterschied besteht also im größten Exponenten von $|\Sigma|$ und $|Z|$, welcher von der jeweiligen Größe von $V$, sowie $I$ und $J$ abhängt. Konkret ergeben sich damit für die Level $R_m$ und $L_m$ mit $m \geq 2$ die Platzbedarfe und Laufzeiten in Tabelle 6.2.

| Level | Musterart | Platzbedarf NL Algorithmus | Laufzeit P Algorithmus |
|---|---|---|---|
| $R_m$ | Teilwortmuster | $O(\log \|Z\|)$ | $O\left(\|Z\|^m \cdot \|\Sigma\|^{J+1}\right)$ |
| | Faktormuster | $O(\log \|Z\|)$ | $O\left(\|Z\|^{2m-2} \cdot \|\Sigma\|^{I+1}\right)$ |
| $L_m$ | Teilwortmuster | $O(\log \|Z\|)$ | $O\left(\|Z\|^m \cdot \|\Sigma\|^{J+1}\right)$ |
| | Faktormuster | $O(\log \|Z\|)$ | $O\left(\|Z\|^{2m-2} \cdot \|\Sigma\|^{I+1}\right)$ |

**Tabelle 6.2:** Vergleich der Verbotsmusterarten in Bezug auf die Komplexität der Algorithmen.

Die benötigte Potenz von $|Z|$ in der Laufzeit ist circa doppelt so groß bei den Algorithmen für Faktorverbotsmuster im Vergleich zu den Algorithmen für die Teilwortmuster. Die Potenz von $|\Sigma|$, also $I$ beziehungsweise $J$, hängt dagegen von der Position des Verbotsmusters im eingegebenen Automaten ab und lässt sich nicht generisch in Abhängigkeit des Levels oder Verbotsmusters angeben.

## 6.3 Zeugenvergleich

Im Fall, dass der eingegebene Automat das Verbotsmuster enthält, stellt sich bei den oben aufgeführten Algorithmen die Frage, welche Funktionen $h$ und $g$ und welche Wörter $p$ beziehungsweise $p_0, \ldots, p_e$ und $q$ als Zeugen gefunden werden und, ob es Unterschiede in der Güte der gefundenen Lösungen zwischen den Algorithmen gibt. Es ist anzumerken, dass Algorithmus 6.1 und Algorithmus 6.2 entsprechende Zeugen nach einem erfolgreichen Durchlauf bereitstellen können, während die nichtdeterministischen Algorithmen dafür nicht alle nötigen Werte am Ende des Durchlaufs gespeichert haben.

Die nichtdeterministischen Algorithmen können entsprechend angepasst werden, um alle Zeugen zu speichern. Dafür muss jedes Mal, wenn ein Wort geraten wird, dieses auch gespeichert werden, statt nur den Zustand zu speichern, der mit diesem erreicht wird. Bei Algorithmus 1 in [10] betrifft dies die Subroutinen in Zeile 3 und 5, die jeweils ein Wort raten, sowie die Kanten, die in Zeile 7 gespeichert werden, für die jeweils ein Wort gesucht wird. Die Algorithmen in Kapitel 5 können so modifiziert werden, dass die Schleife von Zeile 16 bis 27 nicht über alle Beschriftungen iteriert, sondern alle Kanten gleichzeitig speichert und bearbeitet, indem in jeder Iterationen der inneren Schleife in den Zeilen 23 bis 27 eine Beschriftung geraten wird und alle Kanten mit dieser Beschriftung bearbeitet werden. Somit braucht man auch hier jeweils für die

Zustandsvariablen, die in den Zeilen 4, 9 und 10 initialisiert werden und jede gespeicherte Kante ein gespeichertes Wort. Der zusätzlich benötigte Speicherplatz beläuft sich auf $|\circ| + |q| + \sum_{f=0}^{e} |p_e|$ im Fall des Teilwortmuster-Algorithmus und auf höchstens $|E| + |q| + |p|$ im Fall der Algorithmen für die Faktormuster. Die logarithmische Platzschranke wird dadurch nicht verlassen. Bei diesen Algorithmen werden diejenigen Berechnungspfade als erstes terminieren, die direkt die Zeugen $h, g, p, q$ beziehungsweise $h, g, p_0, \ldots, p_e, q$ raten, für welche die Summe der Länge aller Bilder von $X$ unter $g$ und der Längen von $q$ und $p$ beziehungsweise $p_0$ bis $p_e$ minimal ist. Damit bekommt man sowohl bei den NL-Algorithmen für die Faktormuster, als auch bei denen für die Teilwortmuster als Lösung die Zeugen mit der kürzesten Gesamtlänge aller benötigten Wörter.

Die deterministischen Algorithmen berechnen dagegen alle benötigten Wörter gleichzeitig und nicht nacheinander. Dadurch terminieren sie, sobald für eine Funktion $h$ alle drei Bedingungen erfüllt sind. Diese Funktion $h$, die dann als Zeuge fungiert, zeichnet sich dadurch aus, das das Maximum der Längen der zugehörigen Wörter $p$, beziehungsweise $p_0$ bis $p_e$, sowie $q$ und aller Bilder von $X$ unter $g$ minimal ist verglichen mit den anderen möglichen Funktionen. Somit erhält man bei den deterministischen Algorithmen für Teilwort- und Faktorverbotsmuster jeweils die Zeugen als Lösung, welche die Länge des längsten benötigten Wortes minimieren.

Es lässt sich also feststellen, dass die Algorithmen für beide Musterarten eine minimale Lösung finden, wobei die gleichen Minimierungskriterien verwendet werden. Nun bleibt die Frage, wie sich die Größe dieser minimalen Zeugen der verschiedenen Verbotsmusterarten zueinander verhält. Wir zeigen im Folgenden, dass die Lösung der Algorithmen, die nach Teilwortmustern suchen, höchstens so groß ist wie die der entsprechenden Algorithmen für Faktormuster. Dafür zeigen wir, dass das Vorkommen eines Faktormusters in einem Automaten bedingt, dass auch das entsprechende Teilwortmuster an der selben Stelle des Automaten vorkommt. Dies wird erst beispielhaft für das Level **DA** gezeigt und anschließend für die rechten und linken Ecklevel der Trotter-Weil-Hierarchie.

**Lemma 6.1** *Falls das Faktormuster $M_{\mathbf{DA}1} = (V, E, v_s, v_l, v_r)$ mit Variablenmenge $X$, welches in Kapitel 4 definiert wurde, in einem DEA $\mathcal{A} = (Z, \Sigma, \cdot, z_0, F)$ vorkommt mit $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$ als Zeugen, kommt auch das Teilwortmuster $\mathcal{P}_{\mathbf{DA}} = (\mathcal{S}, j \neq k)$ mit $\mathcal{S} = (V', X', \circ)$ aus [10] in $\mathcal{A}$ vor mit Zeugen $h' : \mathcal{S} \to \mathcal{A}^{h'}$ und $g' : X^* \to \Sigma^*$, für die für jede Kante $(u, v) \in \circ$ das Bild ihrer Beschriftung unter $g'$ dem Bild der Beschriftung eines Weges von $h^{-1}(h'(u))$ nach $h^{-1}(h'(v))$ unter $g$ entspricht und $h'(S) \subseteq h(V)$ gilt.*

*Beweis.* Das Faktorverbotsmuster $M_{\mathbf{DA}1}$ ist in Abbildung 4.1 in Kapitel 4 dargestellt und das Teilwort-Verbotsmuster $\mathcal{P}_{\mathbf{DA}}$ auf Seite 10 in [10]. Wenn nun $M_{\mathbf{DA}1}$ in einem DEA $\mathcal{A}$ vorkommt, existieren Abbildungen $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$, die das Faktormuster in $\mathcal{A}$ abbilden, sodass alle Anforderungen erfüllt sind. Wir definieren die folgenden Abbildungen für das Teilwortmuster:

$$h'(j) = h(v_2) \qquad g'(x) = g(y)g(x)$$
$$h'(k) = h(v_3) \qquad g'(A_x) = g(x)$$

Mit diesen Abbildungen gilt, dass $g'(A_x)$ ein Teilwort von $g'(x)$ ist. Außerdem ist das Bild aller Knoten aus $V'$ vom Startzustand von $\mathcal{A}$ aus zu erreichen, wegen $z_0 \cdot p \cdot g(x) = h(v_2) = h'(j)$ und $z_0 \cdot p \cdot g(x)g(x) = h(v_3) = h'(k)$. Zuletzt gilt auch $h'(j) \not\equiv_{\mathcal{A}} h'(k)$, auf Grund von $h'(j) \cdot g(y) \cdot q \in F \iff h'(k) \cdot g(y) \cdot q \notin F$. Dies bedeutet, dass $\mathcal{P}_{\mathbf{DA}}$ in $\mathcal{A}$ enthalten ist. Des Weiteren

gilt $h'(j,k) = h(v_2, v_3) \subseteq h(V)$, sowie $h^{-1}(h'(j)) \cdot g^{-1}(g'(x)) = h^{-1}(h'(j))$, $h^{-1}(h'(j)) \cdot g^{-1}(g'(A_x)) = h^{-1}(h'(k))$ und $h^{-1}(h'(k)) \cdot g^{-1}(g'(x)) = h^{-1}(h'(k))$, was bedeutet, dass die Abbildung des Teilwortmusters in den Automaten innerhalb der Abbildung des Faktormusters liegt. □

**Theorem 6.2** *Falls das Faktormuster $M = (V, E, v_s, v_l, v_r)$ mit Variablenmenge X, das zu einem Ecklevel der Trotter-Weil-Hierarchie gehört, in einem DEA $\mathcal{A} = (Z, \Sigma, \cdot, z_0, F)$ vorkommt mit $h : V \to Z$ und $g : X \to Synt(L(\mathcal{A}))$ als Zeugen, kommt auch das Teilwortmuster $\mathcal{P} = (\mathcal{S}, j \neq k)$ mit $\mathcal{S} = (V', X', \circ)$ aus [10], das zum selben Level der Trotter-Weil-Hierarchie gehört, in $\mathcal{A}$ vor mit Zeugen $h' : \mathcal{S} \to \mathcal{A}^{h'}$ und $g' : X^* \to \Sigma^*$, für die für jede Kante $(u, v) \in \circ$, das Bild ihrer Beschriftung unter $g'$ dem Bild der Beschriftung eines Weges von $h^{-1}(h'(u))$ nach $h^{-1}(h'(v))$ unter g entspricht und $h'(S) \subseteq h(V)$ gilt. Außerdem gilt bei linken Eckleveln $h(v_s) = h'(r)$ und bei rechten Eckleveln $h(v_l) = h'(j)$ und $h(v_r) = h'(k)$.*

*Beweis.* Der Beweis wird induktiv über die Höhe $m$ des Levels geführt. Sei zunächst $m = 2$. Abbildung 6.1 zeigt oben das Faktormuster für $\mathbf{R}_2$, welches nach der Vorschrift in Kapitel 4 erstellt wurde und unten das Teilwortmuster für $\mathbf{R}_2$, welches nach dem Schema auf Seite 21 in [10] erstellt wurde. Zur besseren Unterscheidung werden im Teilwortmuster alle Varieblennamen mit einem Apostroph versehen.

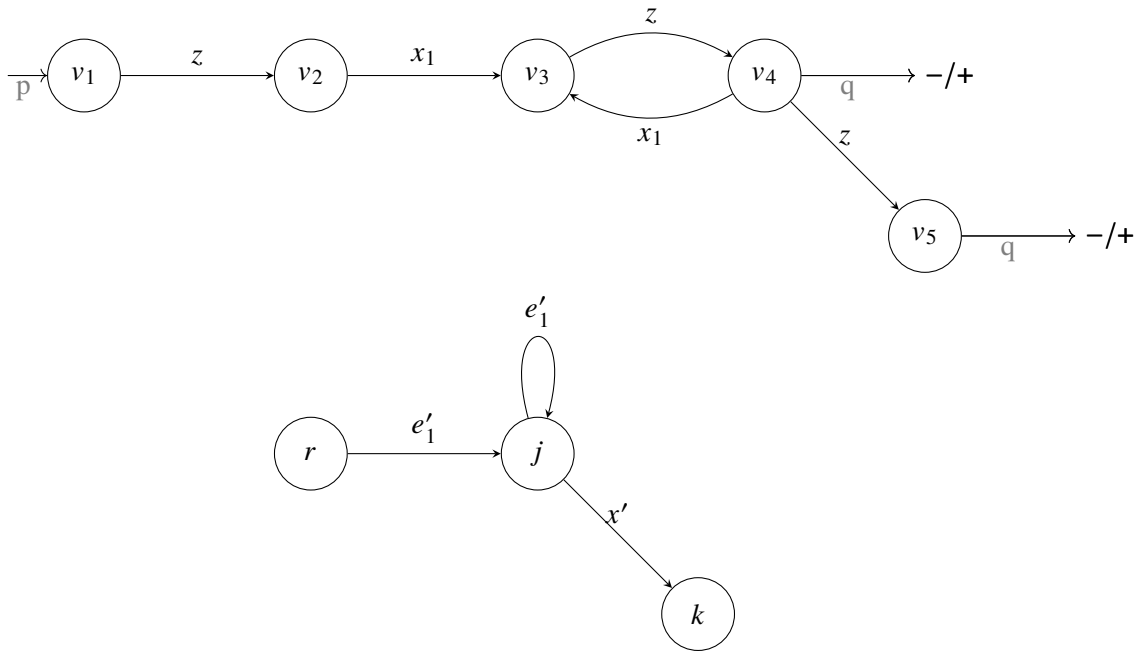



**Abbildung 6.1:** Faktor- und Teilwort-Verbotsmuster für $\mathbf{R}_2$.

Es ist leicht zu sehen, dass die folgenden Abbildungen alle nötigen Anforderungen erfüllen:

$$h'(r) = h(v_2) \qquad g'(x') = g(z)$$
$$h'(j) = h(v_4) \qquad g'(e'_1) = g(x_1)g(z)$$
$$h'(k) = h(v_5)$$

Für $\mathbf{L}_2$ sind die entsprechenden Verbotsmuster in Abbildung 6.2 aufgezeigt, oben das Faktormuster und unten das Teilwortmuster.

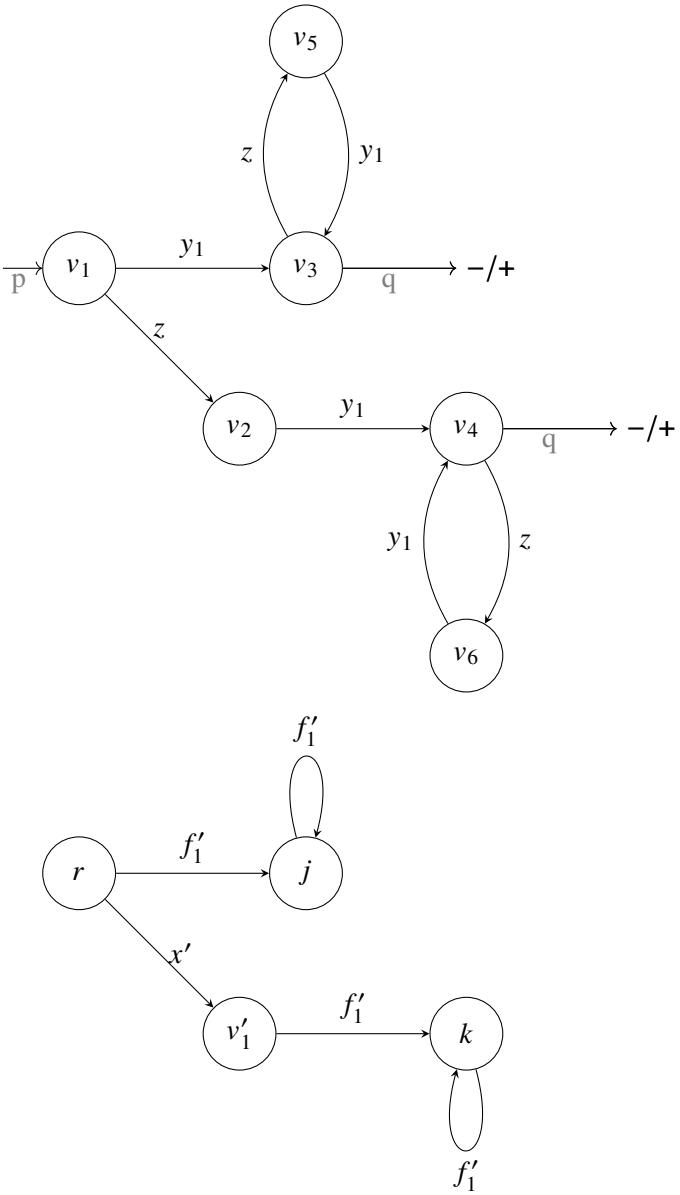



**Abbildung 6.2:** Faktor- und Teilwort-Verbotsmuster für $\mathbf{L}_2$.

Bei diesen Mustern werden die Anforderungen durch die folgenden Abbildungen erfüllt:

$$h'(r) = h(v_1) \qquad g'(x') = g(z)$$
$$h'(v_1') = h(v_2) \qquad g'(f_1') = g(y_1)g(z)$$
$$h'(j) = h(v_5)$$
$$h'(k) = h(v_6)$$

Sei nun $m > 2$. Für den Induktionsschritt nehmen wir an, dass für $i < m$ das Vorhandensein eines Faktormusters für $\mathbf{R}_i$ oder $\mathbf{L}_i$ in einem Automaten zum Vorhandensein des Teilwortmusters dieses Level an der selben Stelle im Automaten führt und wollen diese Aussage nun für $m$ zeigen. Dafür zeigen wir, dass einerseits das Vorkommen des Faktormusters für $\mathbf{L}_m$ zum Vorkommen des Teilwortmusters für $\mathbf{L}_m$ führt und andererseits das Vorkommen des Faktormusters für $\mathbf{R}_m$ zum Vorkommen des Teilwortmusters für $\mathbf{R}_m$ führt. Sei zuerst das Faktormuster für $\mathbf{L}_m$ im DEA $\mathcal{A}$ enthalten mit den Funktionen $h$ und $g$ als Zeugen. Da $\mathbf{L}_m$ in der Trotter-Weil-Hierarchie oberhalb von $\mathbf{R}_{m-1}$ liegt, ist auch das Muster für dieses Level in $\mathcal{A}$ enthalten. In Abbildung 6.3 ist das Grundmuster für $\mathbf{L}_m$ dargestellt, wobei der rote Teil das Grundmuster für $\mathbf{R}_{m-1}$ ist.

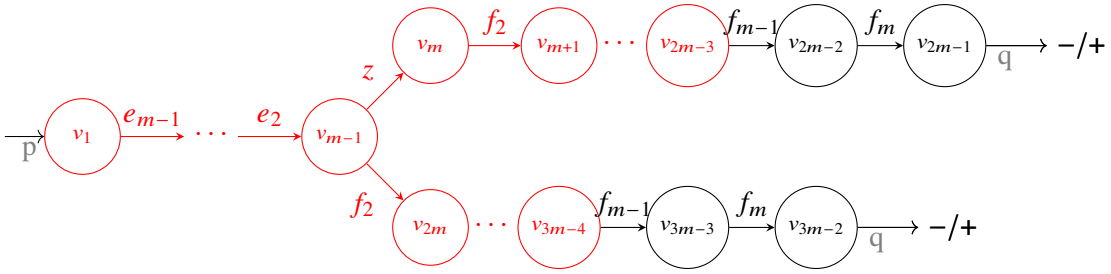



**Abbildung 6.3:** Grundmuster des Faktormusters für $\mathbf{L}_m$ und darin liegendes Grundmuster des Faktormusters für $\mathbf{R}_{m-1}$.

Nach Induktionsvoraussetzung folgt nun, wegen $m - 1 < m$, dass das Teilwortmuster für $\mathbf{R}_{m-1}$ auch in $\mathcal{A}$ enthalten ist, wobei die ausgezeichneten Knoten $j$ und $k$ dieses Musters auf dieselben Zustände des Automaten abgebildet werden, wie die Knoten $v_l$ und $v_r$ des Faktormusters (Knoten $v_{2m-3}$ und $v_{3m-4}$ in Abbildung 6.3). Das Teilwortmuster für $\mathbf{R}_{m-1}$ nennen wir $\mathcal{P}$ und die Zeugen $h'$ und $g'$. Wie in Kapitel 2 angegeben, gilt $\mathbf{L}_m = \mathbf{D}\textcircled{\scriptsize M}\mathbf{R}_{m-1}$. Nach Theorem 22 in [10] folgt, dass das Muster $\mathcal{P}_D$, welches aus $\mathcal{P}$ anhand Definition 16 in [10] gebildet wurde, das Teilwortmuster für $\mathcal{L}_m$ ist. Abbildung 6.4 zeigt das Modifizierungsmuster, um $\mathcal{P}_D$ aus $\mathcal{P}$ zu erhalten. Es bedeutet, dass zwei neue Zustände $j'$ und $k'$ zugefügt werden müssen, die von $j$ und $k$ aus erreicht werden und jeweils eine Schleife haben, wobei alle neuen Kanten mit der selben Variable $f'$ beschriftet sein müssen, für die $f \preceq f'$ für alle Variablen $f$ gelten muss, die in $\mathcal{P}$ vorkommen.

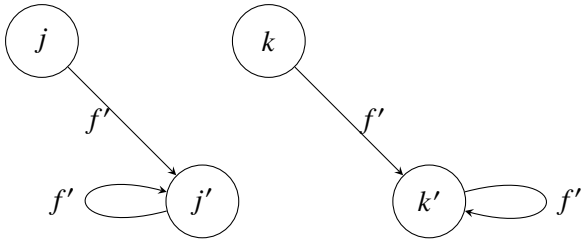



**Abbildung 6.4:** Modifizierungsmuster für ein Malcev Produkt mit **D**.

Um die Verständlichkeit im Folgenden zu vereinfachen, erweitern wir die Darstellung der Funktion $g$, die die Faktormuster für $\mathbf{R}_{m-1}$ und $\mathbf{L}_m$ in $\mathcal{A}$ abbildet so, dass auch die Übergangsvariablen $e_i$ und $f_i$ mit $2 \leq i \leq m$, die im Laufe des Musterausbaus ersetzt werden, von $g$ abgebildet werden können. Das Bild dieser Übergangsvariablen unter $g$ sei dabei die Konkatenation der Bilder der Beschriftungen des kürzesten Wegs, der beim Ersetzen der Kante entsteht, die mit der Übergangsvariable beschriftet ist. Nun werden die Abbildungen $h'$ und $g'$ um folgende Bilder erweitert:

$$h'(j') = h(v_{2m-2}) \qquad g'(f') = g(y_{m-2})g(e_{m-2})\cdots g(e_2)g(z)g(f_2)\cdots g(f_{m-2})$$
$$h'(k') = h(v_{3m-3})$$

Im Einsetzmuster in Abbildung 4.12 kann man sehen, dass diese Wahl von $g'(f')$ tatsächlich die gewünschten gleichbeschrifteten Wege und Schleifen erzeugt. Damit sind $h'$ und $g'$ Zeugen für das Vorkommen von $\mathcal{P}_D$ (also dem Teilwortmuster für $\mathbf{L}_m$) an derselben Stelle im Automaten, sodass $g'(f)$ ein Teilwort von $g'(f')$ für alle Variablen $f$ aus $\mathcal{P}$ ist und $h'(r)$ mit $h(v_s)$ übereinstimmt. Dies erfüllt den Induktionsschritt für linke Ecklevel.

Gleiches soll nun für das entsprechende rechte Ecklevel $\mathbf{R}_m$ gezeigt werden. Sei also das Faktormuster für $\mathbf{R}_m$ im DEA $\mathcal{A}$ enthalten. Da $\mathbf{L}_{m-1}$ in der Trotter-Weil-Hierarchie unterhalb von $\mathbf{R}_m$ liegt, ist auch das Faktormuster für dieses Level in $\mathcal{A}$ enthalten. Die entsprechenden Zeugen für das Enthalten der Faktormuster seien $h$ und $g$. In Abbildung 6.5 ist das Grundmuster für $\mathbf{R}_m$ dargestellt, wobei der rote Teil das Grundmuster für $\mathbf{L}_{m-1}$ ist.

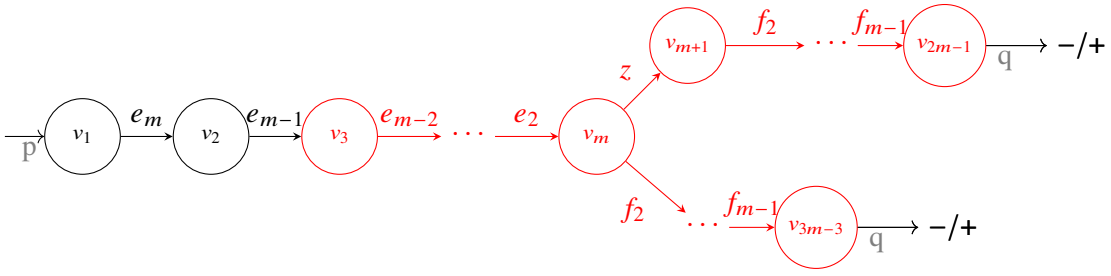



**Abbildung 6.5:** Grundmuster des Faktormusters für $\mathbf{R}_m$ und darin liegendes Grundmuster des Faktormusters für $\mathbf{L}_{m-1}$.

Mit der Induktionsvoraussetzung folgt, dass auch das Teilwortmuster für $\mathbf{L}_{m-1}$ an dieser Stelle in $\mathcal{A}$ enthalten ist mit Zeugen $h'$ und $g'$, wobei insbesondere $g'(r) = g(v_s) = g(v_3)$ gilt. Sei $\mathcal{P}$ dieses Teilwortmuster. Es gilt $\mathbf{R}_m = \mathbf{K}Ⓜ\mathbf{L}_{m-1}$, wie in Kapitel 2 definiert. Nach Theorem 22 in [10] folgt, dass das Muster $\mathcal{P}_K$, welches aus $\mathcal{P}$ anhand Definition 16 in [10] gebildet wurde, das Teilwortmuster für $\mathcal{R}_m$ ist. Das Modifizierungsmuster, welches benutzt wird, um $\mathcal{P}$ zu $\mathcal{P}_K$ zu erweitern, ist in Abbildung 6.6 dargestellt.

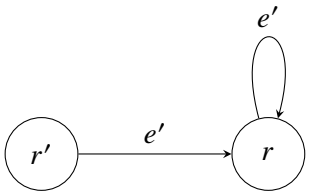



**Abbildung 6.6:** Modifizierungsmuster für ein Malcev Produkt mit $\mathbf{K}$.

Es muss also ein neuer Wurzelknoten zugefügt werden, der eine Kante zum alten Wurzelknoten hat, welche gleich beschriftet ist, wie eine Schleife, die an den alten Wurzelknoten gefügt wird. Für diese Beschriftung $e'$ muss zusätzlich $e \preceq e'$ für alle Beschriftungsvariablen $e$ aus $\mathcal{P}$ gelten.

Mit der gleichen Notations-Erweiterung der Funktion $g$ wie oben, definieren wir folgende zusätzliche Bilder für $h'$ und $g'$:

$$h'(r') = h(v_2) \qquad g'(e') = g(e_{m-2}) \cdots g(e_2)g(z)g(f_2) \cdots g(f_{m-2})g(x_{m-2})$$

Diese Abbildungen $h'$ und $g'$ bezeugen das Vorkommen von $\mathcal{P}_K$ in $\mathcal{A}$, sodass $h'(j) = h(v_l)$ und $h'(k) = h(v_r)$ gilt. Das erfüllt den Induktionsschritt für rechte Ecklevel. □

Damit ist nun gezeigt, dass die Lösung für das Vorhandensein eines Faktormusters, die Algorithmus 5.1 beziehungsweise Algorithmus 6.1 findet, dazu führt, dass Algorithmus 1 in [10] beziehungsweise Algorithmus 6.2 für das selbe Level auf Eingabe des selben Automaten, eine darin enthaltene Lösung ausgibt, sofern es nicht ein noch kleineres Bild des Teilwortmusters gibt. Das Bild eines Faktormusters, das durch die ausgegebenen Zeugen definiert wird ist also mindestens so groß wie das Bild eines Teilwortmusters, das durch die ausgegebenen Zeugen definiert wird.

# 7 Prüfen auf variable Verbotsmuster

Nun werden Algorithmen betrachtet, die Automaten auf Verbotsmuster prüfen, wobei das Verbotsmuster nicht fest sondern Teil der Eingabe ist. Um einen Algorithmus zu konstruieren, der für die gesamte Trotter-Weil-Hierarchie Automaten auf Verbotsmuster prüft, modifizieren wir die Algorithmen in Kapitel 5, Kapitel 6 und [10] so, dass das Muster, auf das geprüft werden soll nicht fester Teil des Algorithmus ist, sondern gemeinsam mit dem zu prüfenden Automaten als Eingabe übergeben wird. Die restliche Funktionsweise der Algorithmen ändert sich dadurch nicht, da weiterhin dieselben Daten zur Verfügung stehen, nur nicht mehr als konstante Werte sondern als variable Eingabe. Die Auswirkung, die dies auf die Platzbedarfe und Laufzeiten hat, wird nun aufbauend auf den Ergebnissen aus Kapitel 6 betrachtet.

Bei den modifizierten nichtdeterministischen Algorithmen ergibt sich für die Faktormuster ein Platzbedarf von $|V| + \max\left\{|\{(u,v) \in E \mid b(u,v) = x\}| \mid x \in X\right\}$. Aus den Mustern in Abbildung 4.11, Abbildung 4.12 und Abbildung 4.13 kann man ableiten, dass die Variable $z$ bei allen Varietäten der Trotter-Weil-Hierarchie diejenige ist, welche die meisten Kanten beschriftet. Damit können wir den Platzbedarf zu $|V| + |\{(u,v) \in E \mid b(u,v) = z\}|$ umschreiben. Für die Teilwortmuster ergibt sich ein Platzbedarf von $|V| + |\circ|$ beim modifizierten nichtdeterministischen Algorithmus. Da die Knoten- und Kantenmengen bei diesen Algorithmen zur Eingabe gehören, sind die Platzbedarfe von $O\big(\log(|V|+|E|)\big)$ für die Faktormuster und $O\big(\log(|V|+|\circ|)\big)$ für die Teilwortmuster logarithmisch in der Eingabegröße.

Im deterministischen Fall ergibt sich für das Finden von Faktormustern durch die modifizierte Version von Algorithmus 6.1 eine Laufzeit von $O\big((|H| \cdot (|V| + |X| + |E|)) + (|H| \cdot |E| \cdot |\Sigma|^{I+1})\big)$, wie aus den Angaben in Kapitel 6 entnommen werden kann. Dies lässt sich zusammenfassen zu $O\big(|Z|^{|V|} \cdot (|E| \cdot |\Sigma|^{I+1} + |V| + |X|)\big)$. Bei diesem allgemeinen Algorithmus sind $|V|$ und $|E|$ keine konstanten Werte, sondern wachsen mit steigendem Level des eingegeben Verbotsmusters. Die Laufzeitangabe benötigt also Teile der Eingabegröße im Exponenten und damit braucht der Algorithmus deterministisch exponentielle Zeit (EXP). Der aus Algorithmus 6.2 konstruierte Algorithmus zum Finden eines eingegebenen Teilwortmusters hat entsprechend Kapitel 6 eine Laufzeit von $O\big((|H| \cdot (|V| + |X| + |\circ|)) + (|H| \cdot |\circ| \cdot |\Sigma|^{J+1}) + (|H| \cdot (|\circ| + |V|) \cdot |\Sigma|^{J})\big)$, was sich zu $O\big(|Z|^{|V|} \cdot (|\Sigma|^{J+1} \cdot |\circ| + |V| + |X|)\big)$ vereinfachen lässt. Auch hier ergibt sich eine Laufzeit in EXP, da $|V|$ im Exponenten vorkommt. Das exponentielle Wachstum des Musters bei steigendem Hierarchie-Level geht nicht in die Laufzeitbetrachtungen ein, da es proportionale Auswirkungen auf die Eingabegröße und die Laufzeit hat.

# 8 Zusammenfassung und Ausblick

Diese Arbeit untersucht Entscheidungskriterien, mit denen man die syntaktischen Monoide von regulären und $\omega$-regulären Sprachen in die Monoidvarietäten der Trotter-Weil-Hierarchie einordnen kann. Die Trotter-Weil-Hierarchie enthält Varietäten zwischen **DA**, der Varietät von Monoiden, deren reguläre $\mathcal{D}$-Klassen aperiodische Halbgruppen sind, und $\mathbf{J_1}$, der Varietät der kommutativen und idempotenten Monoide. Die inneren Varietäten der Hierarchie können als Schnittlevel, Ecklevel und Joinlevel angeordnet werden. **DA**, sowie die Schnittlevel konnten mit den Fragmenten der Quantorenalternierungshierarchie innerhalb von $FO^2$ assoziiert werden [23], [13]. Analog zu Beziehungen bei höheren Klassen, ist zusätzlich zu der Assoziation von Monoidvarietäten und Logikfragmenten von Interesse, ob sich Sprachklassen damit in Verbindung bringen lassen. Da bereits MSO mit den regulären Sprachen zusammenfällt, handelt es sich bei den Sprachklassen, die zur Trotter-Weil-Hierarchie gehören um Teilmengen der regulären Sprachen und damit können diese durch DEAs dargestellt werden. Bei DEAs kann das Konzept der Verbotsmuster verwendet werden. Das sind gerichtete, beschriftete Graphen, die auf Automaten abgebildet werden. In dieser Arbeit wird eine Definition für Verbotsmuster erstellt, die keine Minimalität des untersuchten Automaten fordert.

Rupp hat ein allgemeines Schema zur Konstruktion von Verbotsmustern für die Varietäten der Trotter-Weil-Hierarchie erstellt, sodass für alle DEAs das Nicht-Enthalten eines dieser Verbotsmuster mit der Zugehörigkeit des syntaktischen Monoids seiner erkannten Sprache zu der entsprechenden Varietät äquivalent ist [16]. Damit gibt es jeweils ein Kriterium, dass genau diejenigen Sprachen erfüllen, die zu einer Varietät der Trotter-Weil-Hierarchie gehören. Das Verbotsmusterschema ist in dieser Arbeit in Kapitel 4 aufgeführt und für nicht minimierte Automaten angepasst. Zusätzlich werden in diesem Kapitel explizite Muster für die Monoidvarietäten **DO**, sowie **DA**, **J**, **L** und **R**, welche die Enden der Hierarchie bilden, entwickelt. Hierbei wird für die einzelnen Varietäten auf mehrere $\omega$-Gleichungen, die diese Varietäten charakterisieren, eingegangen und somit eine Übersicht der Verbotsmuster für die Trotter-Weil-Hierarchie aufgebaut. Insbesondere wird für die Varietät **DA** noch ein Muster als passendes Verbotsmuster widerlegt, welches eine naheliegende Wahl wäre. In einer zukünftigen Arbeit könnte nach Widerlegungen für weitere Muster dieser Art gesucht werden.

Da man Logikformeln auf unendlichen genauso wie auf endlichen Wörtern auswerten kann, legt die Verbindung zu den Logikfragmenten nahe, dass neben regulären Sprachen auch $\omega$-reguläre Sprachen für die Trotter-Weil-Hierarchie gefunden werden können. $\omega$-reguläre Sprachen können durch CMAs dargestellt werden, eine Spezialisierung von Büchi-Automaten. Für diese Automaten wird in Kapitel 3 auch eine passende Verbotsmusterdefinition erstellt. Um die syntaktischen Monoide von Sprachen aus $\Sigma^\omega$ besser zu bearbeiten, können diese in je ein fin-syntaktisches und ein inf-syntaktisches Monoid aufgeteilt werden [10]. In dieser Arbeit werden Verbotsmuster, welche die Trotter-Weil-Hierarchie für fin-syntaktische Monoide charakterisieren, definiert. Um die

Hierarchie auch für inf-syntaktische Monoide und damit insgesamt für die syntaktischen Monoide der $\omega$-regulären Sprachen zu charakterisieren, ist noch ein Verbotsmuster für die Varietät **DA** notwendig, das in dieser Arbeit nicht gefunden werden konnte.

Für die entwickelten Verbotsmuster werden in dieser Arbeit Algorithmen konstruiert, die eingegebene Automaten auf das Vorkommen des jeweiligen Musters überprüfen. Dabei werden sowohl nichtdeterministische als auch deterministische Algorithmen konstruiert, wobei für die nichtdeterministischen eine logarithmische Platzschranke gezeigt wird und die deterministischen in Polynomialzeit laufen. Außerdem werden modifizierte Versionen dieser Algorithmen betrachtet, bei denen es nicht ein festes Verbotsmuster für jeden Algorithmus gibt, sondern der Algorithmus beliebige Verbotsmuster der Varietäten der Trotter-Weil-Hierarchie als Eingabe bekommt. Hier erhält man logarithmische Platzbeschränkung für die nichtdeterministischen Algorithmen und exponentielle Zeitbeschränkung für die deterministischen Algorithmen.

Henriksson und Kufleitner definieren entsprechende eingeschränktere Verbotsmuster für die Trotter-Weil-Hierarchie für Sprachen über endlichen und unendlichen Wörtern [10]. Bei diesen eingeschränkten Verbotsmuster werden Teilwortbeziehungen zwischen den Kantenbeschriftungen gefordert. In Kapitel 6 stellen wir einen Vergleich der Verbotsmuster mit und ohne Teilwortbeschränkungen auf. Dabei werden einerseits die Größen der Muster betrachtet und explizit berechnet, wobei bei den Teilwortmustern eine lineare und bei den Faktormustern eine exponentielle Größe in Bezug auf die Höhe der entsprechenden Varietät in der Trotter-Weil-Hierarchie gefunden wird. Andererseits werden die Algorithmen verglichen, welche die verschiedenen Verbotsmuster in Automaten finden. Bei den Algorithmen mit festem Verbotsmuster zeigen sich bei beiden Musterarten vergleichbare Komplexitäten, logarithmischer Platz im nichtdeterministischen Fall und polynomielle Zeit im deterministischen Fall. Die deterministischen Algorithmen mit variablem Muster benötigen exponentielle Zeit. Beim Vergleich der Zeugen, welche die Algorithmen bei erfolgreicher Suche nach einem Verbotsmuster liefern, wird gezeigt, dass alle Algorithmen minimale Lösungen finden. Von besonderem Interesse ist hierbei, dass die Präsenz eines Faktormusters in einem Automaten die Präsenz des, zur gleichen Varietät der Trotter-Weil-Hierarchie gehörenden, Teilwortmusters impliziert, was in Kapitel 6 gezeigt wird. Damit ist das minimale Bild eines Teilwortmusters in einem Automaten maximal so groß wie das minimale Bild eines passenden Faktormusters.

In einer zukünftigen Arbeit könnte die entgegengesetzte Implikation geprüft werden, also inwiefern die Präsenz eines Teilwortmusters in einem Automaten auf die Präsenz eines Faktormusters schließen lässt. Von Interesse könnte auch die Überprüfung der Existenz eines Faktorverbotsmuster, das die Zugehörigkeit von inf-syntaktischen Monoiden zu **DA** charakterisiert, sein. Die Schnittlevel der Trotter-Weil-Hierarchie entsprechen den Voll-Leveln der Quantorenalternierungshierarchie. Diese Quantorenalternierungshierarchie enthält auch Halblevel. Für eine nähere Verknüpfung zur Logik, könnte man für diese Level auch Verbotsmuster konstruieren und diese in einem weiteren Schritt mit den Teilwortverbotsmustern von Henriksson und Kufleitner vergleichen. Letztlich wäre es interessant, ob die Algorithmen aus Kapitel 7 verbessert werden können und, ob es möglich ist, sie so auszubauen, dass für einen eingegebenen Automaten das tiefste Level der Trotter-Weil-Hierarchie bestimmt wird, in dem das syntaktische Monoid der entsprechenden Sprache enthalten ist.

# Literaturverzeichnis


[1] A. Arnold. „A syntactic congruence for rational $\omega$-languages“. In: *Theoretical Computer Science* 39 (1985). Third Conference on Foundations of Software Technology and Theoretical Computer Science, S. 333–335. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(85)90148-3. URL: https://www.sciencedirect.com/science/article/pii/0304397585901483 (zitiert auf S. 23).

[2] J. R. Büchi. „Weak Second-Order Arithmetic and Finite Automata“. In: *Mathematical Logic Quarterly* 6.1-6 (1960), S. 66–92. DOI: https://doi.org/10.1002/malq.19600060105. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/malq.19600060105 (zitiert auf S. 15).

[3] O. Carton, M. Michel. „Unambiguous Büchi automata“. In: *Theoretical Computer Science* 297.1 (2003). Latin American Theoretical Informatics, S. 37–81. ISSN: 0304-3975. DOI: https://doi.org/10.1016/S0304-3975(02)00618-7. URL: https://www.sciencedirect.com/science/article/pii/S0304397502006187 (zitiert auf S. 22).

[4] S. Cho, D. T. Huynh. „Finite-automaton aperiodicity is PSPACE-complete“. In: *Theoretical Computer Science* 88.1 (1991), S. 99–116. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(91)90075-D. URL: https://www.sciencedirect.com/science/article/pii/030439759190075D (zitiert auf S. 25).

[5] V. Diekert, M. Kufleitner, G. Rosenberger. *Arithmetik, Kryptographie, Automaten und Gruppen*. Berlin, Boston: De Gruyter, 2013. ISBN: 9783110312614. DOI: doi:10.1515/9783110312614. URL: https://doi.org/10.1515/9783110312614 (zitiert auf S. 18).

[6] S. Eilenberg. *Automata, Languages and Machines, Vol. B Pure and Applied Mathematics, Vol. 59*. 1976 (zitiert auf S. 19).

[7] C. C. Elgot. „Decision problems of finite automata design and related arithmetics“. In: *Transactions of the American Mathematical Society* 98.1 (1961), S. 21–51 (zitiert auf S. 15).

[8] C. Glaßer, H. Schmitz. „Languages of Dot-Depth 3/2“. In: Bd. 42. Jan. 2000, S. 555–566. ISBN: 978-3-540-67141-1. DOI: 10.1007/s00224-007-9002-0 (zitiert auf S. 25).

[9] V. Henriksson. „Membership and separation problems inside two-variable first order logic“. In: (Feb. 2022). DOI: 10.26174/thesis.lboro.17872106.v1. URL: https://repository.lboro.ac.uk/articles/thesis/Membership_and_separation_problems_inside_two-variable_first_order_logic/17872106 (zitiert auf S. 15).

[10] V. Henriksson, M. Kufleitner. „Forbidden Patterns for $FO^2$ Alternation over Finite and Infinite Words“. In: *International Journal of Foundations of Computer Science* (2023) (zitiert auf S. 15, 40, 41, 43, 45, 48, 49, 52, 55–58, 60, 62, 63, 65, 66).

[11] N. Immerman. „Nondeterministic Space is Closed under Complementation“. In: *SIAM Journal on Computing* 17.5 (1988), S. 935–938. DOI: 10.1137/0217058. eprint: https://doi.org/10.1137/0217058. URL: https://doi.org/10.1137/0217058 (zitiert auf S. 48).

[12] O. Klíma, L. Polák. „Forbidden Patterns for Ordered Automata“. In: *Journal of Automata, Languages and Combinatorics* 25.2–3 (2020), S. 141–169. DOI: 10.25596/jalc-2020-141. URL: https://doi.org/10.25596/jalc-2020-141 (zitiert auf S. 25).

[13] M. Kufleitner, P. Weil. „The FO2 alternation hierarchy is decidable“. en. In: (2012). DOI: 10.4230/LIPICS.CSL.2012.426. URL: http://drops.dagstuhl.de/opus/volltexte/2012/3688/ (zitiert auf S. 65).

[14] R. McNaughton, S. A. Papert. *Counter-Free Automata (MIT research monograph no. 65)*. The MIT Press, 1971 (zitiert auf S. 15).

[15] A. Nerode. „Linear automaton transformations“. In: *Proceedings of the American Mathematical Society* 9.4 (1958), S. 541–544 (zitiert auf S. 15).

[16] T. Rupp. „Verbotsmuster bei deterministischen endlichen Automaten in der Trotter-Weil-Hierarchie“. Bachelorarb. Universität Stuttgart, 2013 (zitiert auf S. 37, 40, 45, 65).

[17] H. Schmitz. „The Forbidden Pattern Approach to Concatenation Hierarchies“. In: (Aug. 2002) (zitiert auf S. 25).

[18] H. Schmitz, K. Wagner. „The Boolean Hierarchy over Level 1/2 of the Straubing-Thérien Hierarchy“. In: (Feb. 1970) (zitiert auf S. 25).

[19] M. P. Schützenberger. „Sur le produit de concaténation non ambigu“. In: *Semigroup forum*. Bd. 13. 1. 1976, S. 47–75 (zitiert auf S. 15).

[20] J. Stern. „Characterizations of some classes of regular events“. In: *Theoretical Computer Science* 35 (1985), S. 17–42. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(85)90003-9. URL: https://www.sciencedirect.com/science/article/pii/0304397585900039 (zitiert auf S. 25).

[21] J. Stern. „Complexity of some problems from the theory of automata“. In: *Information and Control* 66.3 (1985), S. 163–176. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(85)80058-9. URL: https://www.sciencedirect.com/science/article/pii/S0019995885800589 (zitiert auf S. 25).

[22] R. Szelepcsényi. „The method of forced enumeration for nondeterministic automata“. In: *Acta Informatica* 26 (1988), S. 279–284 (zitiert auf S. 48).

[23] D. Thérien, T. Wilke. „Over words, two variables are as powerful as one quantifier alternation“. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. 1998, S. 234–240 (zitiert auf S. 20, 65).

[24] B. A. Trakhtenbrot. „Finite automata and the logic of one-place predicates“. In: *Sibirskii Matematicheskii Zhurnal* 3.1 (1962), S. 103–131 (zitiert auf S. 15).

[25] P. Trotter, P. Weil. „The lattice of pseudovarieties of idempotent semigroups and a non-regular analogue“. In: *Algebra Universalis* 37.4 (1997), S. 491–526 (zitiert auf S. 15).

**Erklärung**

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

---

Ort, Datum, Unterschrift