Institut für Formale Methoden der Informatik

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 61

Verbotsmuster bei deterministischen endlichen Automaten in der Trotter-Weil-Hierarchie

Tobias Rupp

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Volker Diekert
Betreuer/in:	Dr. Manfred Kufleitner
Beginn am:	17. Mai 2013
Beendet am:	16. November 2013
CR-Nummer:	F.4.3

Kurzfassung

Die Klasse **DA** von endlichen Monoiden besitzt eine Vielzahl an Charakterisierungen logischer, kombinatorischer und algebraischer Art. Unter anderem bildet **DA** eine sogenannte Varietät. Um die Struktur der Untervarietäten von **DA** zu untersuchen, betrachteten Trotter und Weil den Schnitt mit idempotenten Halbgruppen. Da deren Hierarchie bereits bekannt war, führte dies zu einer neuen Varietätenhierarchie innerhalb von **DA**, der Trotter-Weil-Hierarchie. Überdies konnten für diese Varietäten algebraische Charakterisierungen angegeben werden, die Entscheidbarkeit ermöglichten.
Falls nun eine Sprache in der Form eines deterministischen endlichen Automaten gegeben ist, soll effizient entschieden werden können, ob das syntaktisches Monoid in einer gegebenen Varietät der Trotter-Weil-Hierarchie liegt. In dieser Bachelorarbeit werden unter Verwendung von Verbotsmustern entsprechende Entscheidungsalgorithmen der Komplexitätsklassen **NL** und **P** konstruiert. Verbotsmuster bezeichnen hierbei Graphkonfigurationen, die im Automatengraph nicht vorkommen dürfen.

Inhaltsverzeichnis

1 Einleitung

Das Konzept der Automaten aus der theoretischen Informatik findet sich in den verschiedensten Bereichen der praktischen Informatik wieder: Beim Compilerbau werden Automaten zur Implementierung von Parsern eingesetzt, Programmabläufe können mit ihnen modelliert werden und auch in der technischen Informatik werden sie bei der Entwicklung neuer Hardware verwendet. Die vermutlich wichtigsten Automaten in der Automatentheorie sind die deterministischen endlichen Automaten (DEA). Verbunden mit den DEAs sind auch die von ihnen erkannten regulären formalen Sprachen. Darüber hinaus lassen sich Teilmengen der regulären Sprachen auch mithilfe von bestimmten logischen Formeln beschreiben, beispielsweise mit der Prädikatenlogik erster Stufe über Wörtern (FO). Wir können also DEAs anhand der Darstellbarkeit ihrer erkannten Sprachen klassifizieren. Ein typisches Anwendungsgebiet stellen Modellprüfungen dar: Gegeben ist ein abstraktes Systemmodell in Form eines Automaten und eine Spezifikation in Form einer logischen Formel. Nun muss überprüft werden, ob die erkannte Sprache des Automaten diese Formel erfüllt. Generell ist es einfacher Formeln von ausdrucksschwächeren Logiken zu überprüfen. Dazu betrachten wir verschiedene Logikfragmente von FO, d.h. eingeschränkte Logiken. Beispielsweise können wir bei der Logik FO die Anzahl der verwendeten Variablen auf zwei (FO^2) und zusätzlich die Anzahl der Wechsel zwischen All- und Existenzquantoren im Syntaxbaum der Formel begrenzen. Mit den Einschränkungen dieser Logik ist typischerweise ein geringerer Aufwand an Berechnungsressourcen verbunden. Nun stellt sich die Frage, ob die Sprache eines Automaten in einem bestimmten Logikfragment darstellbar ist.
Da solche Sprachen ein eindeutiges syntaktisches Monoid besitzen, können wir dieses Problem auch algebraisch angehen. Trotter und Weil entdeckten eine entsprechende algebraische Hierarchie [TW97], welche Kufleitner und Weil mit der Anzahl der Quantorenalternierungen innerhalb von FO^2 in Verbindung bringen konnten[KW12a]. Insbesondere existieren für die Komponenten (Varietäten) der Trotter-Weil-Hierarchie algebraische Entscheidungskriterien, sodass das Problem insgesamt entscheidbar ist.
Wir wollen nun anhand eines gegebenen Automaten diese Problem möglichst schnell entscheiden, d.h. ohne sein syntaktisches Monoid auszurechnen, was bereits exponentiell viel Zeit kosten würde. Dafür wird mithilfe von Verbotsmustern, die an die obigen algebraischen Entscheidbarkeitskriterien angelehnt sind, in dieser Bachelorarbeit ein Polynomialzeitalgorithmus konstruiert. Verbotsmuster bezeichnen hierbei Graphkonfigurationen, die im Automatengraph nicht vorkommen dürfen.

Gliederung

Dazu ist die Arbeit in folgender Weise gegliedert:

Kapitel 2 – Grundlagen: Hier wird eine kurze Einführung über Automaten, die durch sie erkannten Sprachen, und über die im Zusammenhang stehenden syntaktischen Monoiden gegeben. Des weiteren wird der grundsätzlichen Aufbau der Trotter-Weil-Hierarchie geklärt und auf verwandte Strukturen verwiesen.

Kapitel 3 – Muster ist der Hauptteil dieser Bachelorarbeit: Hier werden zunächst die Verbotsmuster definiert und anschließend daran eines für **DA** sowie die Musterbildungsvorschrift für beliebige Varietäten der Trotter-Weil-Hierarchie angeben. Hauptsächlich wird dann gezeigt, dass ein Muster im entsprechenden Minimalautomaten enthalten ist, genau dann wenn sein syntaktisches Monoid nicht in der korrespondierenden Varietät liegt.

Kapitel 4 – Prüfen auf Muster Schließlich wird ein Algorithmus konstruiert, der in nichtdeterministisch-logarithmischem Platz entscheiden kann, ob ein Muster aus Kapitel 3 in einem Automaten enthalten ist. Aufbauend darauf kann ein deterministischer Polynomialzeitalgorithmus erstellt werden, der die Zugehörigkeit des syntaktischen Monoids zu einer Varietät der Trotter-Weil-Hierarchie entscheiden kann.

2 Grundlagen

Dieses Kapitel gibt zunächst eine kurze Zusammenfassung der grundlegenden Automatentheorie bezüglich deterministischen endlichen Automaten und der durch sie erkannten Sprachen. Danach wird das syntaktische Monoid eingeführt. Im Anschluss daran wird noch kurz die wesentliche Struktur der Trotter-Weil Hierarchie angesprochen. Im Allgemeinen werden wir hier unsere Notation festlegen.

2.1 Automaten

Definition 2.1.1. *Sei Σ ein Alphabet, dann ist eine Sprache L eine beliebige Teilmenge von Σ^**

Definition 2.1.2. *Ein deterministischer endlicher Automat, kurz DEA, ist definiert als ein 5-Tupel:*

$$\mathcal{A} = (P, \Sigma, \delta, p_o, F)$$

P[1] bezeichnet die Menge der Zustände, Σ das Eingabealphabet, $p_0 \in P$ der Startzustand und $F \subseteq P$ die Menge der Endzustände, und schließlich ist $\delta : P \times \Sigma \to P$ die Zustandsübergangsfunktion.

Solange klar ist, für was die einzelnen Zeichen stehen, werden wir auch kurz $p_1 a = p_2$ für $\delta(p_1, a) = p_2$ schreiben. Die Sprache $L(A)$ ist die Teilmenge der Wörter aus Σ^*, mittels denen der Automat von p_0 in einen Endzustand überführt wird. Wir bezeichnen das leere Wort als ϵ. Reguläre Sprachen sind alle jene Sprachen L, für die es einen DEA A gibt, sodass $L = L(A)$, d.h. der Automat erkennt die Sprache. Wir werden im Folgenden nur reguläre Sprachen verwenden. Ferner ist bekannt, dass es für jede Sprache L einen bis auf Isomorphie eindeutigen Minimalautomaten A mit $L = L(A)$ gibt. Falls wir im Folgenden von einem nicht näher spezifizierten Automaten sprechen, so meinen wir immer einen Minimalautomaten.

2.2 Syntaktisches Monoid

Zuerst benötigen wir die allgemeine Definition von Monoiden:

Definition 2.2.1. *Ein Monoid ist ein Tripel $(N, \cdot, 1)$. Hierbei ist N ein Menge, $\cdot$ eine zweistellige Verknüpfung $N \times N \to N$ und $1 \in N$ das neutrale Element. Außerdem gelten noch 2 Bedingungen:*

- *$\forall a, b, c \in N$ gilt: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ (Assoziativität)*
- *$\forall a \in N$ gilt: $a \cdot 1 = 1 \cdot a = a$ (neutrales Element)*

[1] Wir verwenden hier P statt dem üblichen Z für Zustände, da wir mit z Variablen bezeichnen.

Solange keine Missverständnisse auftreten, werden wir uns das Verknüpfungssymbol sparen und wir schreiben für $a \cdot b$ auch kürzer ab.

Wir werden nur endliche Monoide betrachten, d.h. Monoide bei denen gilt: $|N| \neq \infty$. Des weiteren sei ein Element $e \in N$ idempotent, falls gilt: $e^2 = e \cdot e = e$.

Wir stellen zunächst fest, dass für jedes $x \in N$ eine positive Zahl n existiert, sodass x^n idempotent ist:

Beweis. [Lau09] Betrachten wir dazu alle möglichen Potenzen von x. Weil B endlich ist muss es $1 \leq i < j$ geben mit $x^i = x^j$. Wir zeigen nun, dass mit $p = j - i$ das Element x^{ip} idempotent ist. Es gilt: $x^i = x^{i+p} = x^j$. Aus der Faktorisierung $x^{i+p} = x^i x^p$, können wir induktiv schließen: $x^{i+kp} = x^j$ für $k \geq 1$. Für $p > 1$ gilt dann $(x^{ip})^2 = x^{ip}x^{ip} = x^{i(p-1)}x^{i+ip} = x^{i(p-1)}x^i = x^{ip}$. Für $p = 1$ gilt: $(x^{ip})^2 = x^{i+ip} = x^i = x^{ip}$ □

Wir nennen das Produkt aller Zahlen, die als Potenz ein Element aus N idempotent machen, ω. Damit gilt für jedes $x \in N : x^\omega = (x^\omega)^2$.

Für das syntaktische Monoid brauchen wir zunächst die Kongruenzrelation $\sim_L$ auf einer Sprache L:

Definition 2.2.2. $\forall x, y \in \Sigma^*$ *gilt:* $x \sim_L y \Leftrightarrow (\forall u, v \in \Sigma^* : uxv \in L \Leftrightarrow uyv \in L)$

Definition 2.2.3. $\mathcal{T}(\mathcal{A}) = \Sigma^* / \sim_L$ *ist das Restklassenmonoid bezüglich* $\sim_L$ *über dem freien Monoid* Σ^**. Wir nennen* $\mathcal{T}(\mathcal{A})$ *das syntaktische Monoid.*

Für einen deterministischen Minimalautomaten gilt zusätzlich: Jedes Element von $\mathcal{T}(\mathcal{A})$ induziert eine Transitionsfunktion $P \to P$ im Automaten, d.h. für $x, y \in \Sigma^*$ und $p \in P$ gilt: Falls $x \sim_L y$, so ist $px = py$.

Beweis. Angenommen, dies wäre nicht der Fall, dann gibt es $x, y \in \Sigma^*$ mit $x \sim_L y$ und $p_1, p_2, p_3 \in Z$ mit $p_1 x = p_2 \neq p_3 = p_1 y$.
Aus $x \sim_L y$ folgt aber auch, dass $\forall u, w \in \Sigma^* : uxw \in L \Leftrightarrow uyw \in L$ und mit Startzustand p_0 auch $p_0 uxw \in F \Leftrightarrow p_0 uyw \in F$, also insbesondere auch $p_1 xw \in F \Leftrightarrow p_1 yw \in F$. Folglich ist für alle $w \in \Sigma^*$: $p_2 w = p_3 w$, dass würde aber bedeuten, dass man die beiden Zustände p_2, p_3 „verschmelzen“ könnte (s. z.B. [HMU07]), was im Widerspruch zur Eigenschaft der Minimalität des Automaten steht.

□

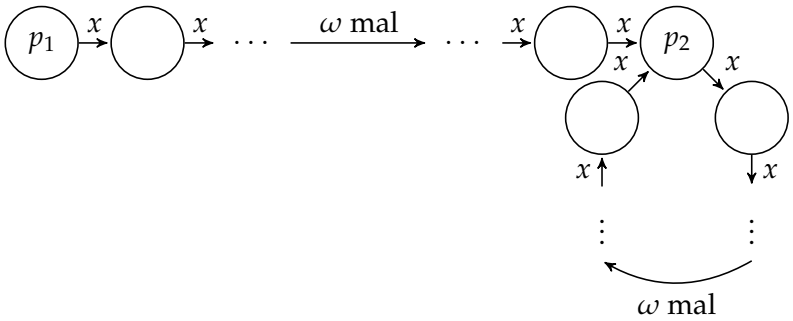
Wir schreiben analog zur Notation mit Zustandsübergängen $p_1\tilde{x} = p_2$ mit $\tilde{x} \in \mathcal{T}(\mathcal{A})$ für die Transitionsfunktion $\tilde{x} : p_1 \to p_2$. Außerdem gilt, alle Elemente von $\mathcal{T}(\mathcal{A})$ müssen (im Minimalautomat) eine andere Transitionsfunktion darstellen, genauer gesagt:

Lemma 2.2.1. *Für $\tilde{x}, \tilde{y} \in \mathcal{T}(\mathcal{A})$ mit $\tilde{x} \neq \tilde{y}$ folgt: Es existieren $p \in P$ mit $p\tilde{x} \neq p_1\tilde{y}$.*

Beweis. Angenommen das wäre nicht der Fall, dann gilt für alle $p \in P$: $p\tilde{x} = p\tilde{y}$. Seien nun $x, y \in \Sigma^*$ zwei Wörter aus den Restklassen von $\tilde{x}$ bzw. $\tilde{y}$. Dann gilt hier:
$\forall p : px = py$, also auch
$\forall\, u \in \Sigma^*$: $p_0ux = p_0uy$ (p_0 sei der Startzustand) und
$\forall\, u, w \in \Sigma^*$: $p_0uxw = p_0uyw$
$\forall\, u, w \in \Sigma^*$: $p_0uxw \in F \Leftrightarrow p_0uyw \in F$ und damit auch
$\forall\, u, w \in \Sigma^*$: $uxw \in L(\mathcal{A}) \Leftrightarrow uyw \in L(\mathcal{A})$ und schließlich:
$x \sim_L y$, was aber im Widerspruch zur Annahme steht, dass x und y zu den verschiedenen Elementen $\tilde{x}$ bzw. $\tilde{y}$ von $\mathcal{T}(\mathcal{A})$ gehören. □

Anders ausgedrückt, werden wir das syntaktische Monoid in Minimalautomaten als Transitionsmonoid verwenden. Damit werden die Hauptbeweise wesentlich vereinfacht.

Eine weitere wichtige Beobachtung, die wir später oft brauchen ist, dass $\forall p_1, p_2 \in P, \forall x \in \mathcal{T}(\mathcal{A})$ gilt: $p_1x^\omega = p_1(x^\omega)^2$ und damit auch für $p_2 = p_1x^\omega$ folgt $p_2 = p_1(x^\omega)^2 = p_2x^\omega$. Anschaulich bedeutet das, es existiert ein Zyklus x^ω im Graphen des Automaten, auf dem p_2 liegt:



2.3 Trotter-Weil-Hierarchie

2.3.1 Green'sche Relationen

Die Green'schen Relationen $\mathcal{R}$ und $\mathcal{L}$ sind ein wichtiges Hilfsmittel zur Beschreibung der Struktur von Monoiden und werden hier verwendet, um die verschiedenen Level der Trotter-Weil-Hierarchie zu definieren.

Definition 2.3.1. *Für ein Monoid* $(N, \cdot, 1)$ *und* $x, y \in N$ *gilt:*

$$x \mathcal{R} y \Leftrightarrow xN = yN$$

$$x \mathcal{L} y \Leftrightarrow Nx = Ny$$

Mit $xN = \{z | z = x \cdot y \text{ mit } y \in N\}$
Anders gesagt, gilt $x \mathcal{R} y$ genau dann, wenn ein z_1 existiert sodass $x = yz_1$ und ein z_2 existiert, sodass $xz_2 = y$ gilt. Für $\mathcal{L}$ gilt analoges mit den z's auf der linke Seite.

Aufbauend auf den Green'schen Relationen definieren wir die Kongruenzen $\sim_K$ und $\sim_D$ auf einem Monoid $(N, \cdot, 1)$.

Definition 2.3.2. *Mit* $x, y \in N$*, sei* $x \sim_K y$*, falls die folgende Implikation für alle Idempotenten* $e \in N$ *erfüllt ist:*

$$ex \mathcal{R} e \vee ey \mathcal{R} e \Rightarrow ex = ey$$

und $x, y \in B$*, sei* $x \sim_D y$*, falls die folgende Implikation für alle Idempotenten* $e \in N$ *erfüllt ist:*

$$xe \mathcal{L} e \vee ye \mathcal{L} e \Rightarrow xe = ye$$

2.3.2 Varietäten

Wir geben hier eine kurze Zusammenfassung nach [KL12]. Eine Varietät [2] ist eine Klasse von endliche Monoiden, welche abgeschlossen unter Bildung von Untermonoiden, homomorphen Abbildungen und endlichen direkten Produkten ist. Seien $\mathbf{V}, \mathbf{W}$ Varietäten, dann bezeichnet der *Join* $\mathbf{V} \vee \mathbf{W}$ die kleinste Varietät, die beide enthält. Der Schnitt $\mathbf{V} \cap \mathbf{W}$ bezeichnet genau die Schnittmenge der beiden Varietäten.

Für den Start der Trotter-Weil Hierarchie definieren wir noch zuerst $\mathcal{R}/\mathcal{L}$-trivial:

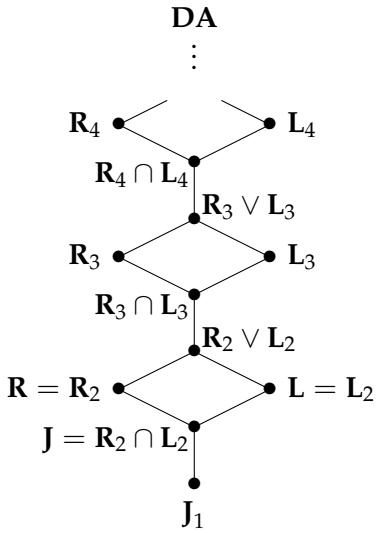
Definition 2.3.3. *$\mathcal{R}/\mathcal{L}$-trivial*
Ein Monoid $(N, \cdot, 1)$ *sei $\mathcal{R}$-trivial falls für alle* $x, y \in N$ *gilt:* $x \mathcal{R} y \Rightarrow x = y$.
Ein Monoid $(N, \cdot, 1)$ *sei $\mathcal{L}$-trivial falls für alle* $x, y \in N$ *gilt:* $x \mathcal{L} y \Rightarrow x = y$.

Die Varietät $\mathbf{R}$ $(bzw.\mathbf{L})$ enthält alle Monoide, die $\mathcal{R}-$ (bzw. $\mathcal{L}$) trivial sind. Mit diesen Varietäten und Kongruenzen können wir nun die Trotter-Weil-Hierarchie definieren:
Seien $\mathbf{R}_2 = \mathbf{R}$ und $\mathbf{L}_2 = \mathbf{L}$. Für $m \geq 2$ sei ein Monoid $(N, \cdot, 1)$ in $\mathbf{R}_{m+1}$ falls $(N, \cdot, 1)/\sim_D$ in $\mathbf{L}_m$ ist. Implizit werden hier sogenannte Malcev-Produkte verwendet. Analog sei es in $\mathbf{L}_{m+1}$ falls $(N, \cdot, 1)/\sim_K$ in $\mathbf{R}_m$ ist.

[2] In der Literatur wird dies u.U. als Pseudovarietät bezeichnet, im Unterschied zu Varietäten, für die unendliche direkte Produkte erlaubt sind

Die *Join*-Level $\mathbf{R}_m \vee \mathbf{L}_m$ bezeichnen wir als $\mathbf{W}_m$. Insbesondere wurde gezeigt, dass $\mathbf{W}_m = \mathbf{R}_m \vee \mathbf{L}_m \subset \mathbf{R}_{m+1} \cap \mathbf{R}_{m+1}$ ist, d.h. die *Join*-Level fallen nicht mit den Schnittleveln einer Ebene höher zusammen. Eine sehr wichtige Varietät ist **DA** für die gilt $\mathbf{DA} = \bigcup_m \mathbf{R}_m = \bigcup_m \mathbf{L}_m$, d.h. unter anderem, dass alle Varietäten der Trotter-Weil-Hierarchie Subvarietäten von **DA** sind. Der Vollständigkeit halber sei erwähnt, dass $\mathbf{J} = \mathbf{R} \cap \mathbf{L}$ ist und $\mathbf{J}_1$ die Varietät der kommutativen Monoide ist, die nur idempotente Elemente enthalten.

Folgendes Diagramm macht die Trotter-Weil-Hierarchie anschaulich:



2.3.3 Omegaterme und Omegagleichungen

Ein einfache Möglichkeit Varietäten darzustellen sind Omegagleichungen. Wir werden in unseren Hauptbeweisen ausschließlich auf sie zurückgreifen, da sie als algebraische Charakterisierungen sich besonders gut eignen mit ihnen zu rechnen.

Bemerkung 2.3.1. *Um von den nun verwendeten Variablen notationell abzugrenzen, werden wir im Folgenden Monoidelemente, insbesondere des syntaktischen Monoids, als Kleinbuchstaben mit Tilde-Zeichen bezeichnen. Zeichen aus dem Automatenalphabet werden wir in den Hauptbeweisen nicht brauchen.*

Definition 2.3.4. *Sei x eine Variable aus einer Variablenmenge Σ, dann ist das neutrale Element 1 und jedes x ein ω-Term. Falls x und y ω-Terme sind, dann sind xy und x^ω ebenfalls ω-Terme.*

Bsp.: Seien x, y Variablen, dann ist $(xy)^\omega x$ ein ω-Term.

Definition 2.3.5. *Seien ℓ und r ω-Terme, dann ist $[\![\, \ell = r \,]\!]$ eine ω-Gleichung.*

Ein Monoid erfüllt eine ω-Gleichung, falls für jede beliebige Belegung der vorkommenden Variablen die Gleichung erfüllt ist, d.h. sich beide ω-Terme zum gleichen Element auswerten. Für die Zuordnung einer ω-Gleichung zu einer Varietät gilt: Genau dann wenn ein Monoid die ω-Gleichung erfüllt, ist es Element der entsprechenden Varietät.
Bsp.: Eine Varietät $\mathbf{V}$ besitzt die zugeordnete ω-Gleichung $[\![\, x^\omega x = x^\omega \,]\!]$. Für ein Monoid $(N, \cdot, 1)$ gilt: Es existiert ein Element $\tilde{x} \in N$ für das gilt $\tilde{x}^\omega \tilde{x} \neq \tilde{x}^\omega$. Beim Belegen der Variable x mit $\tilde{x}$, wird klar dass die ω-Gleichung nicht erfüllt ist, und damit ist $(N, \cdot, 1)$ nicht in $\mathbf{V}$.

Unsere bisher definierten Varietäten haben nach [KL12] folgende Charakterisierungen in ω-Gleichungen:

$$\mathbf{DA} = [\![\, (xy)^\omega x (xy)^\omega = (xy)^\omega \,]\!]$$

$$\mathbf{R}_m = [\![\, e_m ... e_1 z f_1 ... f_{m-1} = e_m .. e_1 f_1 ... f_{m-1} \,]\!] \text{ für } m > 1$$
$$\mathbf{L}_m = [\![\, e_{m-1} ... e_1 z f_1 ... f_m = e_{m-1} .. e_1 f_1 ... f_m \,]\!] \text{ für } m > 1$$
$$\mathbf{W}_m = [\![\, e_m ... e_1 z f_1 ... f_m = e_m .. e_1 f_1 ... f_m \,]\!] \text{ für } m > 0$$
$$e_1 = f_1 = 1$$
$$e_{i+1} = (e_i ... e_1 z f_1 ... f_i x_i)^\omega$$
$$f_{i+1} = (y_i e_i ... e_1 z f_1 ... f_i)^\omega$$

Anzumerken ist hier, dass es noch andere Rekursionsvorschriften zur Bildung der Varietäten der Trotter-Weil-Hierarchie gibt:

$$\mathbf{W}_m = [\![\, U_i = V_i \,]\!] \text{ für } m > 0$$

$$\mathbf{R}_{m+1} = [\![\, (U_i x)^\omega U_i = (U_i x)^\omega V_i \,]\!] \text{ für } m > 1$$
$$\mathbf{L}_{m+1} = [\![\, U_i (y_i U_i)^\omega = V_i (y_i U_i)^\omega \,]\!] \text{ für } m > 1$$

$$U_1 = z, V_1 = 1$$
$$U_{i+1} = (U_i x_i)^\omega U_i (y_i U_i)^\omega$$
$$V_{i+1} = (U_i x_i)^\omega V_i (y_i U_i)^\omega$$

2.3.4 Diverse Charakterisierungen

Da die Varietät **DA** ein sehr gut untersuchtes Forschungsgebiet darstellt, gibt es diverse Charakterisierungen algebraischer, kombinatorischer und logischer Art. Für eine umfassendere Übersicht siehe z.B [DGK08].
Hinsichtlich logischer Charakterisierungen haben Thérien und Wilke [TW98] gezeigt, dass die auf zwei Variablen begrenzte Prädikatenlogik erster Stufe auf Wörtern(FO^2) der Varietät **DA** entspricht.
Des weiteren hat sich herausgestellt, dass die von Trotter und Weil untersuchte Struktur der Untervarietäten von DA ein breites Spektrum an verschiedenen Charakterisierungen besitzt: Ein Band ist eine idempotente Halbgruppe. Unter anderem zeigten Trotter und Weil[TW97] Zusammenhänge zwischen den *Join*-Leveln und der Struktur der Bandhierarchie, welche unabhängig von Birjukov [Bir70], Fennemore [Fen71] und Gerhard [Ger70] klassifiziert wurde.
Kufleitner und Weil zeigten außerdem, dass es verschiedene Möglichkeiten gibt, die einzelnen Stufen der Trotter-Weil-Hierarchie zu erklimmen: Von Malcev-Produkte [KW10] über sogenannte "condensed ranker"[KW12b] bis hin zur Quantorenalternierung innerhalb der FO^2-Logik [KW12a].
Insbesondere letztes Ergebnis ist interessant: Es wurde gezeigt, dass sich eine Sprache in Prädikatenlogik erster Stufe über Wörtern mit nur zwei Variablen und m Quantorenblöcken genau dann darstellen lässt, wenn ihr syntaktisches Monoid in $\mathbf{R}_{m+1} \cap \mathbf{L}_{m+1}$ liegt. Also gilt: Je tiefer das syntaktische Monoid einer Sprache in der Trotter-Weil-Hierarchie liegt, desto einfacher ist die Sprache darstellbar.
Bezüglich der Ranker, die konzeptionell Aussagen über das nächste bzw. vorherige Vorkommen eines bestimmten Buchstaben innerhalb eines Wortes machen, besteht außerdem Verwandtschaft zur eindeutigen Intervall-Logik von Lodaya, Pandya und Shah [LPS08].

3 Muster

Gegeben sind uns aus Abschnitt 2.3.3 die folgenden algebraischen Charakterisierungen der verschiedenen Levels der Trotter-Weil Hierarchie in ω-Notation:

$$\mathbf{R}_m = [\![\, e_m...e_1zf_1...f_{m-1} = e_m..e_1f_1...f_{m-1} \,]\!] \text{ für } m > 1$$
$$\mathbf{L}_m = [\![\, e_{m-1}...e_1zf_1...f_m = e_{m-1}..e_1f_1...f_m \,]\!] \text{ für } m > 1$$
$$\mathbf{W}_m = [\![\, e_m...e_1zf_1...f_m = e_m..e_1f_1...f_m \,]\!] \text{ für } m > 0$$

$$e_1 = f_1 = 1$$
$$e_{i+1} = (e_i...e_1zf_1...f_ix_i)^\omega$$
$$f_{i+1} = (y_ie_i...e_1zf_1...f_i)^\omega$$

Um diese Gleichungen in Muster für Automaten zu übersetzen, nehmen wir folgende modifizierte aber äquivalente Rekursionsvorschrift:

$$\mathbf{R}_m = [\![\, e_m...e_2zf_2...f_{m-1} = e_m..e_2f_2...f_{m-1} \,]\!] \text{ für } m > 2$$
$$\mathbf{L}_m = [\![\, e_{m-1}...e_1zf_2...f_m = e_{m-1}..e_2f_2...f_m \,]\!] \text{ für } m > 2$$
$$\mathbf{W}_m = [\![\, e_m...e_2zf_2...f_m = e_m..e_2f_2...f_m \,]\!] \text{ für } m > 1$$

$$e_2 = (zx_1)^\omega, f_2 = (y_1z)^\omega$$
$$e_{i+1} = (e_i...e_2zf_2...f_ix_i)^\omega$$
$$f_{i+1} = (y_ie_i...e_2zf_2...f_i)^\omega$$

Da sich $e_1$und f_1 zum neutralen Element 1 ergeben, ändern sie nichts an der späteren Auswertung der ω-Termen und nach der oberen Rekursionsvorschrift gilt somit: $e_2 = (e_1zf_1x_i)^\omega = (1z1x_1)^\omega$ = $(zx_1)^\omega$ und ebenso $f_2 = (y_1e_1zf_1)^\omega = (y_11z1)^\omega = (y_1z)^\omega$. Also können wir zur Bildung von $\mathbf{W}_m$ mit $m > 1$ und für $\mathbf{R}_m, \mathbf{L}_m$ mit $m > 2$ auch diese Rekursionsvorschrift benutzen.
Wir geben hier noch explizit $\mathbf{R}_2$ und $\mathbf{L}_2$ an:

$$\mathbf{R}_2 = [\![\, e_2z = e_2 \,]\!] \quad \mathbf{L}_2 = [\![\, zf_2 = f_2 \,]\!]$$

Analog zu diesen ω-Gleichungen konstruieren wir uns nun induktiv Verbotsmuster für Automaten, deren syntaktisches Monoid diese Gleichungen nicht erfüllen.

3.1 Musterdefinitionen

An dieser Stelle ist wichtig zu bemerken, dass wir die Muster in diesem Kapitel nur für Minimalautomaten definieren, d.h. ein Automat $\mathcal{A}$ sei immer ein Minimalautomat und nach Abschnitt 2.2 dürfen wir Elemente des syntaktischen Monoids als Zustandstransformationen auffassen.

Definition 3.1.1. *Ein Muster M sei ein gerichteter und kantenbeschrifteter Graph $M = (V, E, b, v_s, v_\ell, v_r)$. Hierbei ist V unsere Menge an Knoten, $E \subset V^2$ die Menge an Kanten, $b : E \to X$ eine Beschriftungsfunktion, die jeder Kante eine Variable zuweist, v_s ein bestimmter Startknoten, und $v_\ell, v_r \in V$ mit $v_\ell \neq v_r$ zwei unterschiedliche Knoten. Falls für einen DEA $\mathcal{A} = (P, \Sigma, \delta, p_0, F)$ die Abbildungen $h : V \to P$, mit $h(v_\ell) \neq h(v_r)$ und $g : X \to \mathcal{T}(\mathcal{A})$ existieren, so dass für alle Kanten $(v_1, v_2) \in E$ mit einer Beschriftung $u = b(v_1, v_2)$ gilt: $h(v_1) \cdot g(u) = h(v_2)$, so enthält $\mathcal{A}$ das Muster M.*

Anzumerken ist, dass v_s prinzipiell nicht notwendig ist, sondern nur als Hilfsmittel für einen späteren Beweis dient.

Da wir induktiv die Muster aufbauen müssen, werden wir Einsetzmuster definieren mithilfe denen wir bestehende Muster erweitern werden:

Definition 3.1.2. *$M_e = (V_e, E_e, b_e, v_{ea}, v_{ef})$, hier ist V_e unsere Menge an Knoten, $E_e \subset V_e^2$ die Menge an Kanten und $b_e : E \to X$ wie oben die Beschriftungsfunktion. $v_{ea} v_{ef} \in V_e$ legen fest, wie das Muster eingefügt wird.*

Beim Einsetzen eines Einsetzmusters in ein Muster wählen wir aus dem Muster, in das wir einsetzen wollen, eine Kante (v_a, v_f) aus und ersetzen diese durch unser Einsetzmuster wobei die Knoten v_{ae} und v_{af} die Anknüpfpunkte an v_a bzw. v_f darstellen (anschauliches Beispiel folgt in Abschnitt 3.3).

Formaler haben wir unser (Ursprungs-)Muster $M_u = (V_u, E_u, b_u, v_{s_u}, v_{l_u}, v_{r_u})$ und unser Einsetzmuster $M_e = (V_e, E, b_e, v_{ea}, v_{ef})$. Wenn wir nun M_e in M_u für die Kante $(v_a, v_f) \in E_u$ einsetzen, so erhalten wir ein neues Muster $M_n = (V_n, E_n, b_n, v_{s_n}, v_{\ell_n}, v_{r_n})$ mit:

- $V_n = V_u \cup V_e \backslash \{v_{ae}, v_{af}\}$ (Anmerkung: Die Knotenmengen von verschiedenen Mustern/Einsetzmustern sind natürlich disjunkt)

- $E_n = E_u \backslash \{(v_a, v_f)\} \cup E'_e$ wobei für E'_e gilt:

  $$\forall (v_1, v_2) \in E_e \begin{cases} (v_a, v_f) \in E'_e \text{ falls } v_1 = v_{ae} \wedge v_2 = v_{af} \text{ sonst} \\ (v_a, v_2) \in E'_e \text{ falls } v_1 = v_{ae} \text{ sonst} \\ (v_1, v_f) \in E'_e \text{ falls } v_2 = v_{af} \text{ sonst} \\ (v_1, v_2) \in E'_e \end{cases}$$

  Es werden also einfach bei den Kanten, die die Anknüpfpunkte v_{ae} oder v_{af} enthalten, diese durch v_a bzw. v_f ersetzt.

  Ähnlich verhält es sich bei den Beschriftungen dieser Kanten:

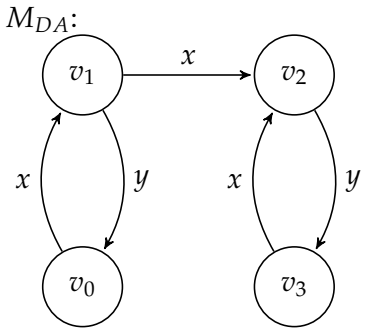
- $$\forall (v_1, v_2) \in E_n : b_n(v_1, v_2) = \begin{cases} b_u(v_1, v_2) \text{ falls } v_1, v_2 \in V_u \text{ sonst} \\ b_e(v_{af}, v_{ef}) \text{ falls } v_1 = v_a \wedge v_2 = v_f \text{ sonst} \\ b_e(v_{ea}, v_2) \text{ falls } v_1 = v_a \wedge v_2 \in V_u \text{ sonst} \\ b_e(v_1, v_{ef}) \text{ falls } v_1 \in V_u \wedge v_2 = v_f \text{ sonst} \\ b_e(v_1, v_2) \text{ hier gilt: } v_1, v_2 \in V_e \end{cases}$$

  Man beachte, dass in M_u und in M_e keine Mehrfachkanten existieren. Da beim Bilden von M_n eine Kante zwischen v_a und v_f entfernt wird und höchstens eine dazukommt, falls es eine Kante (v_{ae}, v_{ef}) gibt, kann M_n ebenfalls keine Mehrfachkanten enthalten.

- $v_{s_n} = v_{s_u}$
- $v_{\ell_n} = v_{\ell_u}$
- $v_{r_n} = v_{r_u}$

3.2 Verbotsmuster für DA

Bevor wir unsere komplizierten, mittels Einsetzen aufgebauten Muster behandeln, betrachten wir zunächst das Verbotsmuster für die Varietät **DA**:



mit $v_s = v_0$, $v_\ell = v_0$ und $v_r = v_3$.

Wir zeigen nun, dass genau dann wenn ein Automat $\mathcal{A}$ das Muster M_{DA} enthält, sein syntaktisches Monoid $\mathcal{T}(\mathcal{A})$ nicht in **DA** liegt.

Lemma 3.2.1. *$\mathcal{A}$ enthält $M_{DA} \Leftrightarrow \mathcal{T}(\mathcal{A}) \notin$ **DA***

Beweis.
$\mathcal{A}$ enthält $M_{DA} \Rightarrow \mathcal{T}(\mathcal{A}) \notin$ **DA**:
Angenommen $\mathcal{A}$ enthält M_{DA}, dann existieren Abbildungen geeignete h, g und es gibt ein Element aus $\mathcal{T}(\mathcal{A})$, nämlich $(\tilde{x}\tilde{y})^\omega$, das $h(v_0)$ in $h(v_0)$ überführt. Analog muss es ein Element der Form $(\tilde{x}\tilde{y})^\omega\tilde{x}(\tilde{x}\tilde{y})^\omega$ geben, das $h(v_0)$ in $h(v_2)$ überführt. Da die Zielknoten v_0 und v_2 nach Voraussetzung auf unterschiedliche Zustände abgebildet worden sind, müssen $(\tilde{x}\tilde{y})^\omega$ und $(\tilde{x}\tilde{y})^\omega\tilde{x}(\tilde{x}\tilde{y})^\omega$ auch unterschiedliche Elemente des syntaktischen Monoids darstellen. Damit ist $[\![\, (xy)^\omega \neq (xy)^\omega x (xy)^\omega \,]\!]$ verletzt und es folgt $\mathcal{T}(\mathcal{A}) \notin$ **DA**.

$\mathcal{A}$ enthält $M_{DA} \Leftarrow \mathcal{T}(\mathcal{A}) \notin$ **DA**:
Die Idee ist, zu zeigen, dass es Zustände (darunter zwei unterschiedliche) gibt, die sich mit passenden Elemente aus $\mathcal{T}(\mathcal{A})$ gemäß des Musters überführen lassen.
Angenommen $\mathcal{T}(\mathcal{A}) \notin$ **DA**, dann existieren $\tilde{x}, \tilde{y} \in \mathcal{T}(\mathcal{A})$ für die gilt: $(\tilde{x}\tilde{y})^\omega \neq (\tilde{x}\tilde{y})^\omega\tilde{x}(\tilde{x}\tilde{y})^\omega$.
Also gibt es nach Lemma 2.2.1 einen Hilfszustand p_h mit $p_h(\tilde{x}\tilde{y})^\omega \neq p_h(\tilde{x}\tilde{y})^\omega\tilde{x}(\tilde{x}\tilde{y})^\omega$.
Nun legen wir nun die benötigten Zustände p_0, p_1, p_2 und p_3 fest:

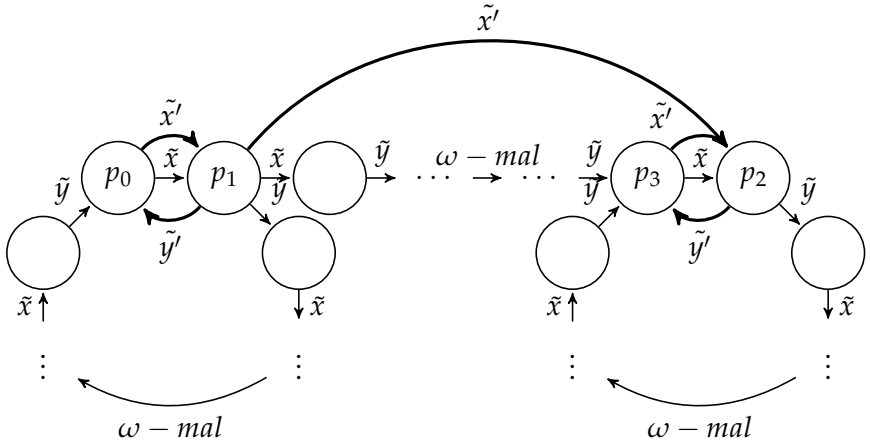
$$p_0 = p_h(\tilde{x}\tilde{y})^\omega, p_1 = p_0\tilde{x}, p_3 = p_1(\tilde{x}\tilde{y})^\omega, p_2 = p_3\tilde{x}$$

Da wir wissen, dass gilt $p_h(\tilde{x}\tilde{y})^\omega = p_h(\tilde{x}\tilde{y})^{2\omega}$ wegen $(\tilde{x}\tilde{y})^\omega = (\tilde{x}\tilde{y})^{2\omega}$, folgt: $p_h(\tilde{x}\tilde{y})^{2\omega} = p_0(\tilde{x}\tilde{y})^\omega = p_0$, p_0 liegt also auf einem Zyklus $(\tilde{x}\tilde{y})^\omega$.

Damit gilt auch:

$$
\begin{aligned}
p_h(\tilde{x}\tilde{y})^\omega &\neq p_h(\tilde{x}\tilde{y})^\omega \tilde{x}(\tilde{x}\tilde{y})^\omega \\
p_0(\tilde{x}\tilde{y})^\omega &\neq p_0(\tilde{x}\tilde{y})^\omega \tilde{x}(\tilde{x}\tilde{y})^\omega \\
p_0 &\neq p_0\tilde{x}(\tilde{x}\tilde{y})^\omega \\
p_0 &\neq p_1(\tilde{x}\tilde{y})^\omega \\
p_0 &\neq p_3
\end{aligned}
$$

Ferner gilt $p_3 = p_1(\tilde{x}\tilde{y})^\omega = p_1(\tilde{x}\tilde{y})^{2\omega} = p_3(\tilde{x}\tilde{y})^\omega$, d.h. p_3 liegt ebenfalls auf einem Zyklus $(\tilde{x}\tilde{y})^\omega$. Anschaulich haben wir folgendes Diagramm ohne die $\tilde{x'}$- und $\tilde{y'}$-Kanten gegeben:



Wir behaupten nun, die Abbildungen h, g mit $h(v_0) = p_0, h(v_1) = p_1, h(v_2) = p_2, h(v_3) = p_3$ und $g(x) = \tilde{x'} = (\tilde{x}\tilde{y})^\omega\tilde{x}$ und $g(y) = \tilde{y'} = y(\tilde{x}\tilde{y})^{\omega-1}$ induzieren das Muster M_{DA}, Man beachte, dass bereits $h(v_l) = h(v_0) = p_0 \neq p_3 = h(v_3) = h(v_r)$ gilt. Überdies kann $v_s = v_0$ gewählt werden, der als v_s bezeichnete Knoten stellt anschaulich den Startpunkt dar, von dem aus die ω-Terme der ω-Gleichung als Pfade aufgefasst zu v_r bzw. v_ℓ führen.
Es verbleibt nun nur noch die Korrektheit der Kanten zu zeigen:

1. $h(v_0)g(x) = h(v_1)$
2. $h(v_1)g(y) = h(v_0)$
3. $h(v_1)g(x) = h(v_2)$
4. $h(v_2)g(y) = h(v_3)$
5. $h(v_3)g(x) = h(v_2)$

1. $p_0(\tilde{x}\tilde{y})^\omega = p_0 \wedge p_0\tilde{x} = p_1 \Rightarrow$
   $p_0(\tilde{x}\tilde{y})^\omega\tilde{x} = p_1 \Rightarrow$
   $p_0x' = p_1 \Rightarrow$
   $h(v_0)g(x) = h(v_1)$

2. $p_0(\tilde{x}\tilde{y})^\omega = p_0 \wedge p_0\tilde{x} = p_1 \Rightarrow$
   $p_1\tilde{y}(\tilde{x}\tilde{y})^{\omega-1} = p_0 \Rightarrow$
   $p_1\tilde{y}' = p_0 \Rightarrow$
   $h(v_1)g(y) = h(v_0)$

3. $p_1 = p_1(\tilde{x}\tilde{y})^\omega = p_3 \wedge p_3\tilde{x} = p_2 \Rightarrow$
   $p_1\tilde{x} = p_1(\tilde{x}\tilde{y})^\omega\tilde{x} = p_3\tilde{x} = p_2 \Rightarrow$
   $p_1\tilde{x}' = p_2 \Rightarrow$
   $h(v_1)g(x) = h(v_2)$

4. $p_3(\tilde{x}\tilde{y})^\omega = p_3 \wedge p_3\tilde{x} = p_2 \Rightarrow$
   $p_2\tilde{y}(\tilde{x}\tilde{y})^{\omega-1} = p_3 \Rightarrow$
   $p_2y' = p_3 \Rightarrow$
   $h(v_2)g(y) = h(v_3)$

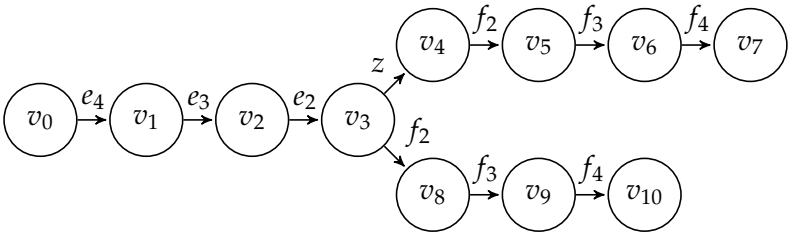
5. $p_3(\tilde{x}\tilde{y})^\omega = p_3 \wedge p_3\tilde{x} = p_2 \Rightarrow$
   $p_3(\tilde{x}\tilde{y})^\omega\tilde{x} = p_2 \Rightarrow$
   $p_3x' = p_2 \Rightarrow$
   $h(v_3)g(x) = h(v_2)$

□

3.3 Musterbeispiel

Dargestellt sind Ursprungmuster $M_{W_{4,4}}$ und Einsetzmuster M_{f_4}, diese stehen in Analogie zu der ω-Gleichung von $\mathbf{W}_4$ bzw. zum ω-Term von f_4.

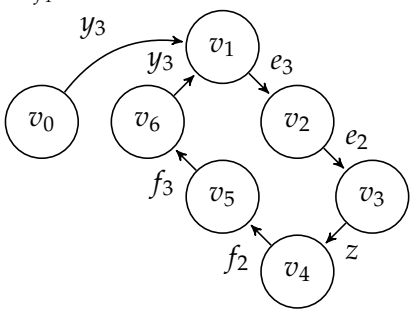
$M_{W_{4,4}}$:



mit $v_s = v_0$, $v_\ell = v_7$ und $v_r = v_{10}$.

Wir wollen dieses und ähnliche Muster später in der folgenden Semantik verwenden: Angenommen wir sind in dem Zustand auf den v_0 abgebildet wird und wenden die linke Seite der ω-Gleichung (ℓ) an, dann erreichen wir den Zustand, auf den v_7 abgebildet wird, analoges gilt für die rechte Seite und v_{10}. Falls der Automat dieses Muster enthält und die beide Seiten der Gleichung in unterschiedliche Zustände führen, kann die Gleichung nicht gelten.
Anmerkung: Wir verwenden die leicht modifizierte Rekursionsgleichung, da wir so auf $e_1 = f_1 = 1$ verzichten können, die wir sonst als ϵ-Kanten realisieren müssten, d.h. wir würden sonst zwei Knoten v und v' mit $b(v, v') = 1$ erzeugen, von denen wir fordern müssten, dass sie immer auf den gleichen Zustand abgebildet werden würden.

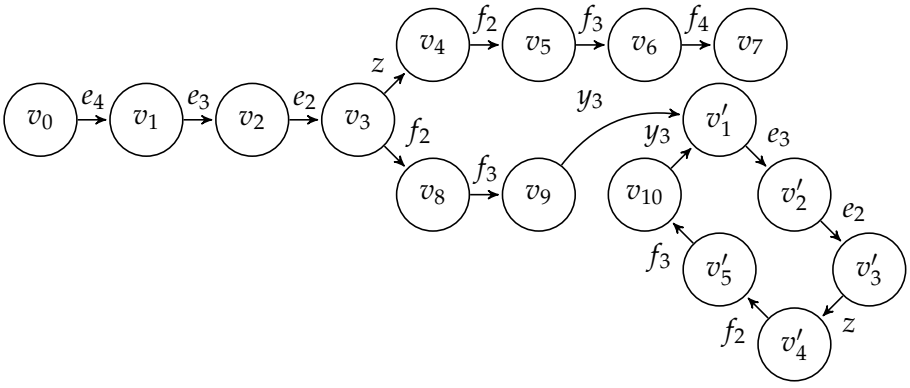
M_{f_4}:



mit $v_{ea} = v_0$ und $v_{ef} = v_6$.

Dieses Muster liegt nahe, da es den Pfad $f_4 = (y_3e_3e_2zf_2f_3)^\omega = y_3(e_3e_2zf_2f_3y_3)^{\omega-1}e_3e_2zf_2f_3$ von v_{ea} nach v_{ef} gibt.

Wenn wir nun das Muster M_{f_4} für die Kante (v_9, v_{10}) einsetzen (also $v_a = v_9, v_f = v_{10}$), die mit f_4 beschriftet ist, erhalten wir folgendes Muster:

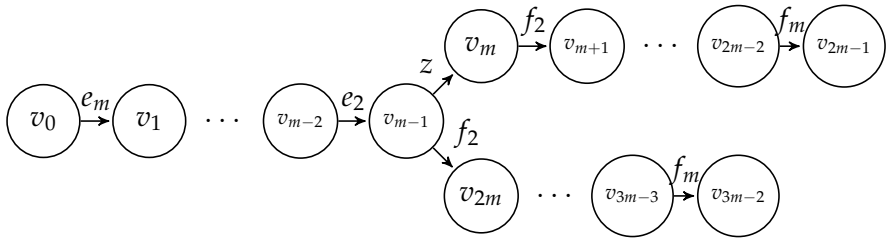


mit $v_s = v_0$, $v_\ell = v_7$ und $v_r = v_{10}$, die von $M_{W_{3,3}}$ übernommen wurden

3.4 Rekursive Muster

Für das *Join*-Level $\mathbf{W}_m$ haben wir zunächst das Muster $M_{W_{m,m}}$, wobei zu beachten ist, dass dieses noch nicht das finale Verbotsmuster darstellt:

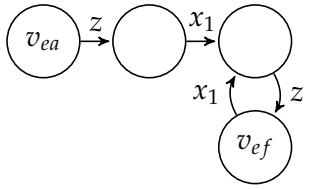
$M_{W_{m,m}}$:



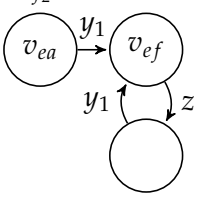
mit $v_s = v_0$, $v_\ell = v_{2m-1}$ und $v_r = v_{3m-2}$.

Passend zur Rekursionsvorschrift haben wir die Einsetzmuster $M_{e_{i+1}}$ und $M_{f_{i+1}}$. Ebenso M_{e_2} und M_{f_2}, die keine Kanten e_i oder f_i besitzen und somit das Rekursionsende darstellen.

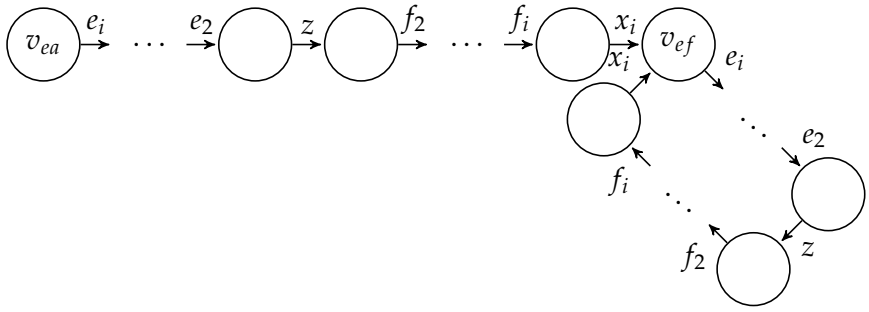
M_{e_2}:

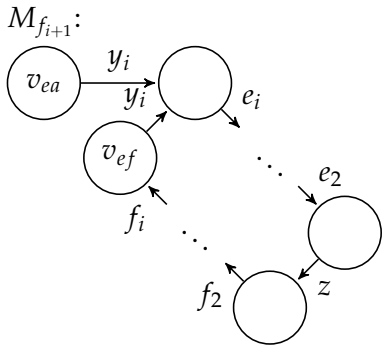


M_{f_2}:



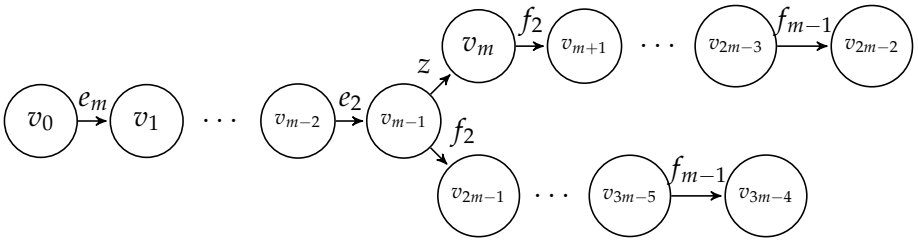
$M_{e_{i+1}}$:





Falls wir für die Varietät $\mathbf{R}_m$ ein Verbotsmuster konstruieren wollen, starten wir mit $M_{R_{m,m}}$.

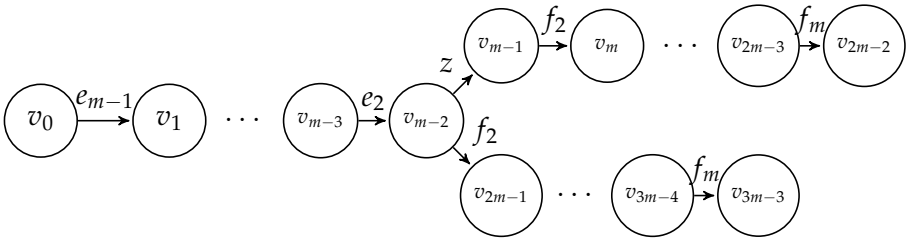
$M_{R_{m,m}}$:



mit $v_s = v_0$, $v_\ell = v_{2m-2}$ und $v_r = v_{3m-4}$.

Für die Varietät $\mathbf{L}_m$ entsprechend $M_{L_{m,m}}$.

$M_{L_{m,m}}$:



mit $v_s = v_0$, $v_\ell = v_{2m-2}$ und $v_r = v_{3m-3}$.

3.5 Äquivalenzbeweis

In diesem Abschnitt wollen wir die Äquivalenz zwischen ω-Gleichung und Muster beweisen: Zunächst bemerken wir, dass wir immer für einen konkreten Minimalautomaten A argumentieren, deswegen steht sein syntaktisches Monoid $\mathcal{T}(\mathcal{A})$ fest und damit auch der konkrete Wert von ω.
Wir führen die Beweise anhand der Muster für die *Join*-Level $\mathbf{W}_m$ durch, da klar wird, dass die Beweise für die Muster von $\mathbf{R}_m$ bzw. $\mathbf{L}_m$ nahezu identisch sind. Für die folgenden Beweise benötigen wir noch einige Definitionen:

Definition 3.5.1. *induktive Muster:*
$M_{W_{m,m}}$ ist bereits definiert. Sei $M_{W_{m,i}}$ das Muster, das entsteht, wenn in $M_{W_{m,i+1}}$ alle Kanten mit Beschriftung e_{i+1} und f_{i+1} durch die Einsetzmuster $M_{e_{i+1}}$ bzw. $M_{f_{i+1}}$ ersetzt werden. Wir schreiben hierzu auch $M_{W_{m,i}} = M_{W_{m,i+1}}[e_{i+1}/M_{e_{i+1}}, f_{i+1}/M_{f_{i+1}}]$.

Für die formale Definition des Einsetzen von Mustern für Kanten s. Abschnitt 3.1. Bei diesen Mustern gibt i immer den höchsten Index an, die eine $e-$ oder f-Kante haben kann. Wir bemerken, dass unser gesuchtes Muster das Muster $M_{W_{m,1}}$ ist, bei dem nur noch Kanten mit Beschriftungen der Form z, x_j, y_j mit $j \in \{1, .., m\}$ vorkommen. $M_{W_{m,0}}$ sei nicht definiert, da in $M_{W_{m,1}}$ keine ersetzbaren Kanten mit Beschriftungen e_1 oder f_1 existieren.

Da es im folgenden Beweis wichtig wird, die Zerlegung eines Elements zu kennen, definieren wir eine Zerlegungsdarstellung.

Definition 3.5.2. *Zerlegungsdarstellung:*
Falls wir Elemente aus $\mathcal{T}(\mathcal{A})$ für die Variablen in einen ω-Term ℓ einsetzen, ohne ihn auszuwerten, nennen wir diese Darstellung $\overline{\ell}$.

Bsp.: Sei $\ell = (xy)^\omega x$ ein ω-Term, und wir belegen die Variable x mit dem Element $\tilde{x}$, sowie y mit $\tilde{y}$, dann ist $\overline{\ell} := (\tilde{x}\tilde{y})^\omega \tilde{x}$.

Wir wollen nun zunächst eine Richtung der Äquivalenz beweisen: Wenn $\mathcal{T}(\mathcal{A}) \notin \mathbf{W}_m$ ist, dann enthält der Automat $\mathcal{A}$ zwangsläufig das Muster $M_{W_{m,1}}$.
Falls $\mathcal{T}(\mathcal{A}) \notin \mathbf{W}_m = [\![\, \ell = r \,]\!]$, dann existieren Belegungen aus $\mathcal{T}(\mathcal{A})$, für die Variablen z, x_j, y_j mit $j \in \mathbb{N}$, sodass die Gleichung nicht erfüllt ist. Wir nennen das Element, das die Variable X belegt, $\tilde{X}$. Wir haben nun die Zerlegungsdarstellungen $\overline{\ell}$ und $\overline{r}$ nach Definition 3.5.2 für die ω-Terme ℓ, r. Wir setzen zusätzlich noch die Elemente $\tilde{e}_2 = (\tilde{z}\tilde{x}_i)$, $\tilde{f}_2 = (\tilde{y}_i\tilde{z})$ sowie $\tilde{e}_{i+1} = (\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i\tilde{x}_i)^\omega$ und $\tilde{f}_{i+1} = (\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega$.
Des weiteren bezeichnen wir mit $(\overline{\ell} \neq \overline{r})_{m,1}$ das Zerlegungspaar $\overline{\ell}, \overline{r}$. $(\overline{\ell} \neq \overline{r})_{m,i+1}$ sei das Zerlegungspaar, das entsteht, wenn im Zerlegungspaar $(\overline{\ell} \neq \overline{r})_{m,i}$ alle Vorkommen von $(\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i\tilde{x}_i)^\omega$ zum Element $\tilde{e}_{i+1}$ und $(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega$ zu $\tilde{f}_{i+1}$ invers zur Rekursionsvorschrift der ω-Gleichungen zusammengefasst werden. Also soll insbesondere gelten:
$(\overline{\ell} \neq \overline{r})_{m,m} = \tilde{e}_m...\tilde{e}_1 z \tilde{f}_1...\tilde{f}_m \neq \tilde{e}_m...\tilde{e}_1\tilde{f}_1...\tilde{f}_m$. Wir schreiben auch:
$(\overline{\ell} \neq \overline{r})_{m,i} = (\overline{\ell} \neq \overline{r})_{m,i+1}[\tilde{e}_{i+1}/(\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i\tilde{x}_i)^\omega, \tilde{f}_{i+1}/(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega]$.

Nun wollen wir über Induktion von m aus das folgende Lemma beweisen:

Lemma 3.5.1. *Falls ein Zerlegungspaar $(\bar{\ell} \neq \bar{r})_{m,i}$ existiert, dann gibt es Abbildungen h, g mit $g(u) = \tilde{u}\ \forall u \in z, e_2..e_i, f_2..f_i$, sodass $\mathcal{A}$ aufgrund von h und g Muster $M_{W_{m,i}}$ enthält.*

$g(u) = \tilde{u}$ schreibt vor, dass die Abbildung g eine Variable u (hier als Beschriftung einer Kante) genau auf das Element, das die Variable u der ω-Gleichung belegt, abgebildet wird. Man beachte, dass die Abbildung der restlichen Variablen keinen Einschränkungen unterliegt. Falls die Abbildungen g, h die Bedingung $h(v_1)g(u) = h(v_2)$ für alle Kanten aus dem Muster M mit $b(v_1, v_2)$ erfüllen, so ist M aufgrund von g und h enthalten.

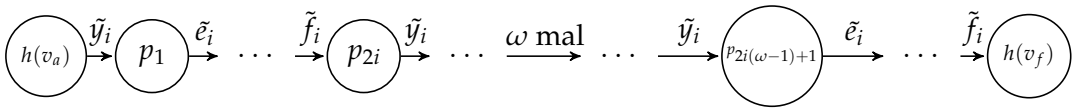
Die Idee bei folgender Induktion ist nun, das Muster $M_{W_{m,i+1}}$, das bereits auf den Automaten abgebildet wurde, zum komplexeren Musters $M_{W_{m,i}}$ umzubauen. Dazu benützen wir das Wissen, dass die Elemente e_i und f_i kanonisch abgebildet werden und aufgrund unserer Konstruktion analog zur Rekursionsvorschrift zerlegbar sind. Da wir unsere Muster ausgehend vom (Ursprungs-)Muster $M_{W_{m,m}}$ konstruieren, starten wir bei m und führen die Induktion durch bis das finale Muster $M_{W_{m,1}}$ erreicht ist.

Beweis. **IA:** Für i=m haben wir das Zerlegungspaar $(\bar{\ell} \neq \bar{r})_{m,m}$, d.h. $\tilde{e}_m...\tilde{e}_2\tilde{z}\tilde{f}_2...\tilde{f}_m \neq \tilde{e}_m...\tilde{e}_2\tilde{f}_2...\tilde{f}_m$. Damit muss es Zustände $p_0, .., p_{3m-2}$ geben, sodass $p_{2m-1} = p_{2m-2}\tilde{f}_m = .. = p_0\tilde{e}_m..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_m \neq p_0\tilde{e}_m..\tilde{e}_2\tilde{f}_2..\tilde{f}_m = .. = p_{3m-2}$ gilt. Damit existiert die kanonische Abbildung $h(v_k) = p_k, k \in \{0, ..., 3m-2\}$, mit $p_{2m-1} = h(v_l) \neq h(r) = p_{3m-2}$ sowie $h(v_s) = p_0$, sodass mit $g(u) = \tilde{u}\ \forall u \in \{z, e_2..e_m, f_2..f_m\}$ aufgrund von h und g das Muster $M_{W_{m,m}}$ in $\mathcal{A}$ enthalten ist.

IS: Nach Induktionsannahme gilt: Falls ein Zerlegungspaar $(\bar{\ell} \neq \bar{r})_{m+1,i}$ existiert, dann gibt es Abbildungen h, g mit $g(u) = \tilde{u}\ \forall u \in \{z, e_2..e_{i+1}, f_2..f_{i+1}\}$, sodass $\mathcal{A}$ aufgrund von g und h Muster $M_{W_{m,i+1}}$ enthält.
Gegeben ist nun die leicht veränderte Darstellung von $(\bar{\ell} \neq \bar{r})_{m,i+1}$, nämlich: $(\bar{\ell} \neq \bar{r})_{m,i}$= $(\bar{\ell} \neq \bar{r})_{m,i+1}[\tilde{e}_{i+1}/(\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i\tilde{x}_i)^\omega, \tilde{f}_{i+1}/(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega]$ Betrachte nun eine Kante aus $M_{W_{m,i}}$ mit $b(v_a, v_f) = f_{i+1}$ wobei v_a, v_f aus V sind. Also gilt $h(v_a)g(f_{i+1}) = h(v_f)$ und wegen unserer kanonischen Abbildung g für f_{i+1} auch $h(v_a)\tilde{f}_{i+1} = h(v_f)$. Da $\tilde{f}_{i+1} = (\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega$ gilt, gilt ebenso $h(v_a)(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega = h(v_f)$. Also gibt es auch Zustände $p_0, p_1, ..p_{(2i)\omega} \in P$ im Automaten für die gilt:

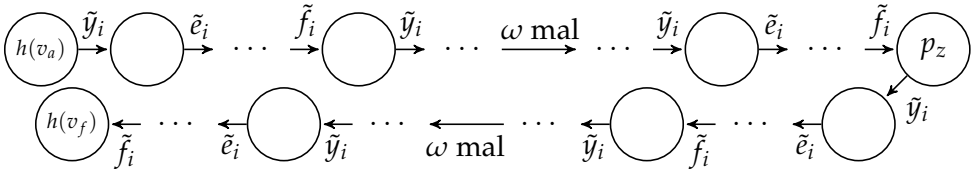
$$\begin{aligned}
h(v_a)(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega &= p_0(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega \\
&= p_1\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^{\omega-1} \\
&= ... \\
&= p_{2i}(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^{\omega-1} \\
&= ... \\
&= p_{(2i)(\omega-1)+1}(\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i) \\
&= p_{(2i)\omega} \\
&= h(v_f)
\end{aligned}$$

Also gibt es im Automaten folgende Struktur zwischen $h(v_a)$ und $h(v_f)$:

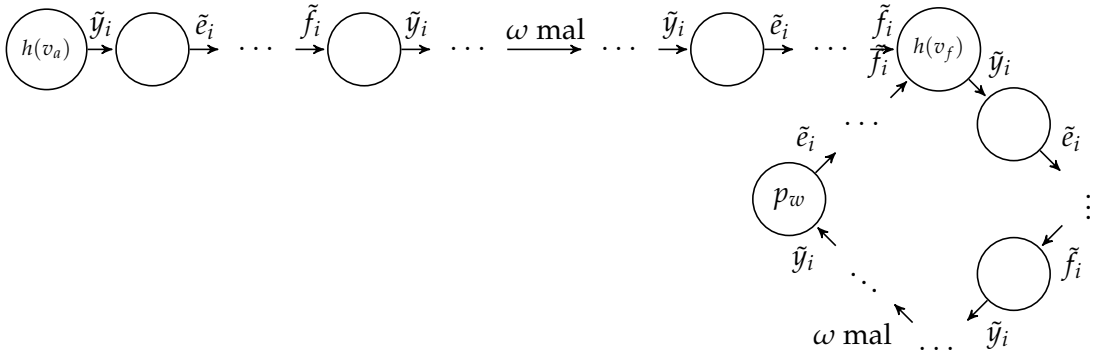


In dieser Darstellung seien die Knoten die Zustände und eine Kante mit Beschriftung eines Elements aus $\mathcal{T}(\mathcal{A})$, die von diesem Element induzierte Zustandstransformation zwischen den beiden Zuständen. In den folgenden Diagrammen werden wir der Übersichtlichkeit halber die explizite Benennung aller Zustände fallen lassen.

Da $(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega = (\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^{2\omega}$ ist, folgt wegen $h(v_a)(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega = h(v_f)$ auch $h(v_a)(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^{2\omega} = h(v_f)$. D.h. im Automaten zwischen $h(v_a)$ und $h(v_f)$ gibt es auch folgende Struktur:



Der Zustand p_z sei definiert als $h(v_a)(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^{2\omega} = p_z(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega = h(v_f)$.
Wir wissen nun, dass $h(v_a)(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega = p_z$ und $h(v_a)(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^{2\omega} = h(v_f)$. Wegen $h(v_f) = h(v_a)(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^{2\omega} = h(v_a)(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^\omega = p_z$, müssen p_z und $h(v_f)$ den gleichen Zustand darstellen, also gibt es im Automaten folgende Struktur:



Mit $p_w = h(v_f)(\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^{\omega-1}\tilde{y}_i$

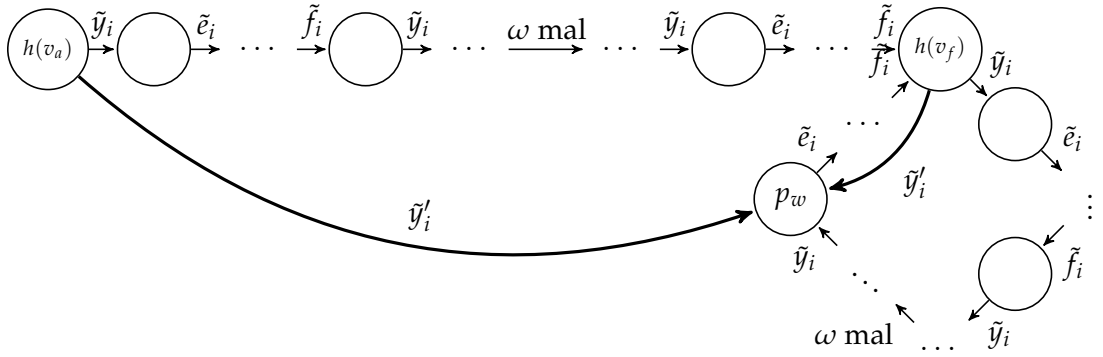
Setze nun $\tilde{y_i'} = (\tilde{y_i}\tilde{e_i}..\tilde{e_2}\tilde{z}\tilde{f_2}..\tilde{f_i})^{2\omega-1}\tilde{y_i}$.
Da $h(v_a)(\tilde{y_i}\tilde{e_i}..\tilde{e_2}\tilde{z}\tilde{f_2}..\tilde{f_i})^{\omega} = h(v_f)$ und $p_w = h(v_f)(\tilde{y_i}\tilde{e_i}..\tilde{e_2}\tilde{z}\tilde{f_2}..\tilde{f_i})^{\omega-1}\tilde{y_i}$ gilt, ist auch

$$p_w = h(v_a)(\tilde{y_i}\tilde{e_i}..\tilde{e_2}\tilde{z}\tilde{f_2}..\tilde{f_i})^{2\omega-1}\tilde{y_i} = h(v_a)\tilde{y_i'}$$

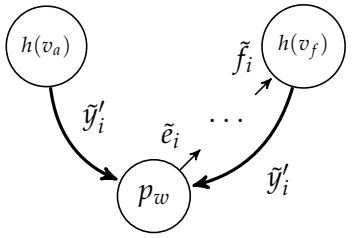
.
Wegen $h(v_f)(\tilde{y_i}\tilde{e_i}..\tilde{e_2}\tilde{z}\tilde{f_2}..\tilde{f_i})^{\omega} = h(v_f)$ ist auch

$$p_w = h(v_f)(\tilde{y_i}\tilde{e_i}..\tilde{e_2}\tilde{z}\tilde{f_2}..\tilde{f_i})^{\omega-1}\tilde{y'} = h(v_f)(\tilde{y_i}\tilde{e_i}..\tilde{e_2}\tilde{z}\tilde{f_2}..\tilde{f_i})^{2\omega-1}\tilde{y} = h(v_f)y_i'$$

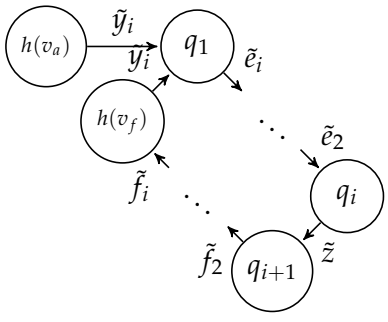
.
Diese beiden y_i'-Transitionen sind im nächsten Bild fett eingezeichnet:



Betrachten wir nun nur noch die folgende Teilstruktur:

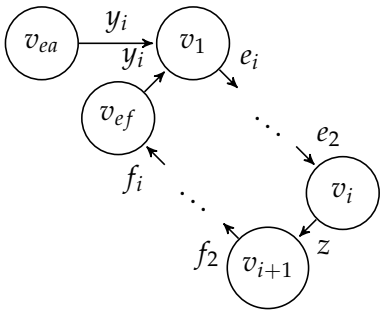


Optisch anders angeordnet, ausführlicher dargestellt und mit Knotennamen q bezeichnet:



Also gilt hier für die Zustände $q \in Q : q_1 = p_w$, $q_2 = q_1\tilde{e}_i$, $q_i = q_1\tilde{e}_i ... \tilde{e}_2$ usw. Für $i = 1$ fallen natürlich alle e und f-Kanten weg und es existieren nur $q_1, h(v_a)$ und $h(v_f)$.

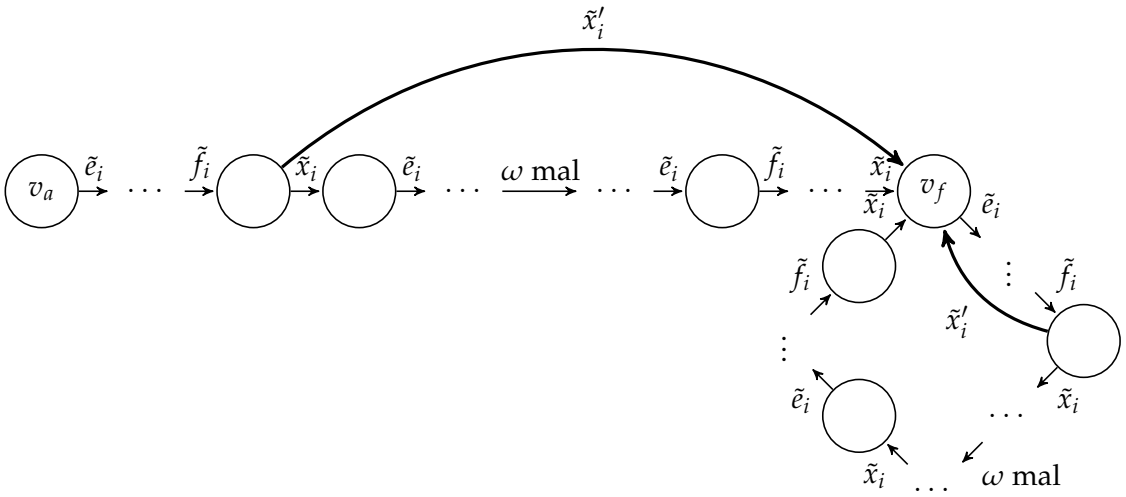
Wir stellen nun fest das diese Struktur im Automaten isomorph zum Einsetzmuster $M_{f_{i+1}}$ ist:



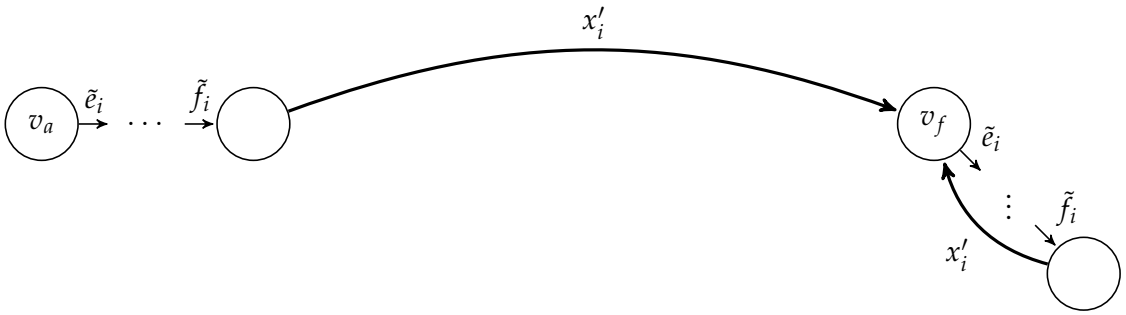
Beim Einsetzen von $M_{f_{i+1}}$ für die Kante (v_a, v_f) in $M_{W_{m,i}}$, kommen die Knoten $v_1, ..v_{2i-1}$ dazu. Wir nennen dieses Muster M'. Die Idee ist nun zeigen, dass M' genauso im Automaten liegt wie $M_{W_{m,i}}$, mit Ausnahme der Kante (v_a, v_f) und des dafür eingesetzten Musters M'.
Da wir nach Induktionsannahme davon ausgehen können, dass es Abbildungen g und h gibt, sodass $M_{W_{m,i}}$ enthalten ist, können wir jetzt h um $h(v_k) = q_k$ für $k \in \{1, .., 2i-1\}$ erweitern. Die Abbildung g erweitern wir lediglich um die bisher noch nicht verwendete Variable y_i mit $g(y_i) = y_i'$. Außerhalb des von uns betrachteten Automatenausschnitts gilt immer noch für alle Kantenbeschriftungen $u = b(v_1, v_2)$ von M': $h(v_1)g(u) = h(v_2)$, da dies auch für $M_{W_{m,i}}$ galt. Angesichts der obigen Isomorphie gilt diese Gleichung innerhalb ebenso, da die Variablen $z, e_2, ..e_i, f_2, ...f_i$ von $M_{f_{i+1}}$ nach Induktionsannahme kanonisch auf $\tilde{z}, \tilde{e}_2, ..\tilde{e}_i, \tilde{f}_2, ...\tilde{f}_i$ abgebildet werden und $g(y_i) = y_i'$ gewählt wurde. Also ist auch M' enthalten.
Auf M' aufbauend können wir nun in analoger Weise alle weiteren Kanten mit Beschriftung f_{i+1} durch $M_{f_{i+1}}$ ersetzen. Dabei müssen wir lediglich die Abbildung h erweitern, da wir immer das gleiche $y_i' = (\tilde{y}_i\tilde{e}_i..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_i)^{2\omega-1}\tilde{y}_i$ verwenden müssen und schon $g(y_i) = y_i'$ gesetzt haben. Nach Ersetzung aller f_i-Kanten erhalten wir schließlich das Muster $M_{m,i+1}[f_{i+1}/M_{f_{i+1}}]$.

Nach der Ersetzung der Kanten f_{i+1} durch $M_{f_{i+1}}$ können wir aufbauend auf $M_{m,i+1}[f_{i+1}/M_{f_{i+1}}]$ wegen $e_{i+1} = (e_i..e_2zf_2..f_ix_i)^\omega$ mit ähnlicher Argumentation die Kanten e_{i+1} durch das Muster $M_{e_{i+1}}$ ersetzen.

Dafür sei $\tilde{x}' = \tilde{x}_i(\tilde{e}_i..\tilde{e_2}\tilde{z}\tilde{f}_2..\tilde{f}_i\tilde{x}_i)^\omega$ und folgende Struktur im Automaten zwischen zwei abgebildeten Knoten mit $l(v_a, v_f) = e_{i+1}$ gegeben:



Außerdem die enthaltene Teilstruktur welche isomorph zu $M_{e_{i+1}}$ ist:



Somit können wir auch alle Kanten mit Beschriftung e_{i+1} von $M_{m,i+1}[f_{i+1}/M_{f_{i+1}}]$ mit $M_{e_{i+1}}$ ersetzen und erhalten so $M_{m,i+1}[e_{i+1}/M_{e_{i+1}}, f_{i+1}/M_{f_{i+1}}] = M_{m,i}$. Da wir nicht verlangen, dass x_i oder y_i von g kanonisch abgebildet werden und wir immer eine geeignete Abbildung h mitführen, gilt:
Falls ein Zerlegungspaar $(\bar{\ell} \neq \bar{r})_{m,i}$ existiert, dann gibt es Abbildungen h, g mit $g(u) = \tilde{u}$ $\forall u \in z, e_2..e_i, f_2..f_i$, sodass $\mathcal{A}$ aufgrund von h und g Muster $M_{W_{m,i}}$ enthält.

□

Da wir für ein beliebige Belegung der Variablen von $\mathbf{W}_m$, die die ω-Gleichung nicht erfüllt, Lemma 3.5.1 anwenden können, können wir folgern:

Lemma 3.5.2. *$\mathcal{T}(\mathcal{A}) \notin W_m \Rightarrow \mathcal{A}$ enthält $M_{W_{m,1}}$*

Nun wollen wir die andere Richtung zeigen, d.h. wenn das Muster enthalten ist, soll folgen, dass die entsprechende ω-Gleichung nicht erfüllt ist.

Definition 3.5.3. *Sei ℓ ein ω-Term dann bezeichnet $\ell' = \ell[X_1/X_2X_3]$ den ω-Term der durch Substituieren aller Vorkommen der Variable X_1 durch den ω-Term X_2X_3 ersetzt wurde, hierbei sind X_2, X_3 ebenfalls Variablen.*

Da wir immer einen konkreten Automaten $\mathcal{A}$ untersuchen, steht wie schon erwähnt sein syntaktisches Monoid fest, und damit auch der Wert ω, also können wir im nächsten Beweis ω-Terme auch als Pfade interpretieren und umgekehrt.

Zunächst definieren wir analog zu den Mustern passende ω-Gleichungen:

Definition 3.5.4. *induktive ω-Gleichungen:*
Analog zu den Mustern soll gelten $\mathbf{W}_{m,m} := [\![\ell = r]\!]$ mit $\ell = e_m...e_2zf_2...f_m$ und $r = e_m...e_2f_2...f_m$, wobei e_j und f_j mit $j \in \{2..m\}$ Variablen wie z seien. Sei außerdem $\mathbf{W}_{m,i} = \mathbf{W}_{m,i+1}[e_{i+1}/(e_i..e_2zf_2..f_ix_m)^\omega, f_{i+1}/(y_me_i..e_2zf_2..f_i)^\omega]$, d.h die Substitution aller vorkommenden Variablen e_{i+1} und f_{i+1} durch $(e_i..e_2zf_2..f_ix_m)^\omega$ bzw. $(y_me_i..e_2zf_2..f_i)^\omega$ in l bzw. r von $\mathbf{W}_m$.

Analog zu den Mustern gibt hier i den größten Index an den eine $e-$ oder f-Variable haben kann. Folglich ist die ω-Gleichung $\mathbf{W}_{m,1}$ äquivalent zu unser schließlich gesuchten ω-Gleichung $\mathbf{W}_m$. $\mathbf{W}_{m,0}$ sei darüber hinaus nicht definiert, da in $\mathbf{W}_{m,1}$ die Variablen e_1 bzw. f_1 nicht existieren.

Nun wollen wir das folgende Lemma zeigen aus dem schließlich auch die spezifische Aussage für $\mathbf{W}_m$=$\mathbf{W}_{m,1}$ und $M_{W_{m,1}}$ folgt.

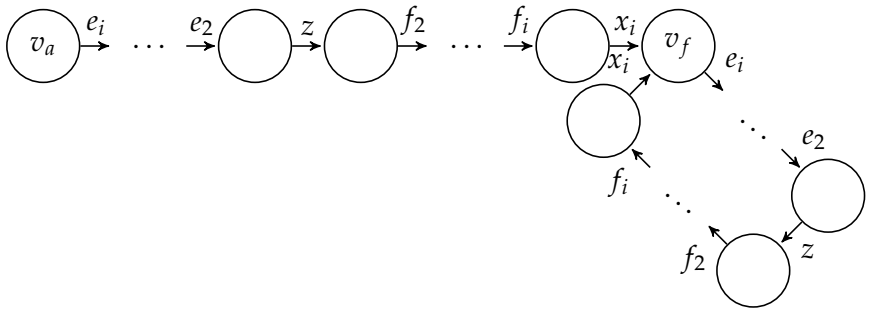
Lemma 3.5.3. *A enthält $M_{W_{m,i}} \Rightarrow \mathcal{T}(\mathcal{A}) \notin \boldsymbol{W}_{m,i}$*

Beweis. Dazu zeigen wir die stärkere Aussage:
A enthält $M_{W_{m,i}} \Rightarrow \mathcal{T}(\mathcal{A}) \notin \mathbf{W}_{m,i} = [\![\ell = r]\!] \wedge$ für $M_{W_{m,i}}$ gilt $v_s \xrightarrow{\ell} v_\ell \wedge v_s \xrightarrow{r} v_r$

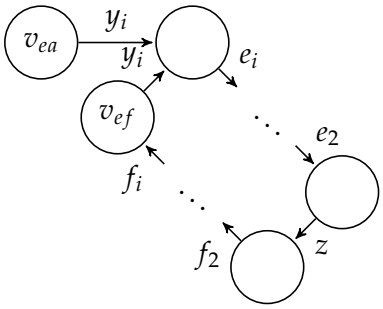
Die Idee ist nun zu zeigen, dass es Pfade in den Mustern gibt, die genau den ω-Termen entsprechen und zu Knoten führen, die auf verschiedene Zustände abgebildet werden. Falls das Muster enthalten ist, gibt es also Belegungen der Pfadvariablen, sodass diese in verschiedene Zustände führen und eben diese Belegungen für die ω-Terme müssen sich deswegen zu verschiedenen Elementen auswerten.

IA: Für i=m, haben wir $M_{m,m}$, d.h. wir haben bereits $v_s \xrightarrow{e_m..e_2zf_2..f_m} v_\ell \wedge v_s \xrightarrow{e_m..e_2f_2..f_m} v_r$ (s. $M_{m,m}$). Da das Muster enthalten ist, gibt es auch Abbildungen h, g mit $h(v_s)g(e_m..e_2zf_2..f_m) = h(v_\ell)$ und $h(v_s)g(e_m..e_2f_2..f_m) = h(v_r)$ mit $h(\ell) \neq h(r)$. Bei kanonischer Benennung der belegenden Elemente gilt also: $\tilde{e}_m..\tilde{e}_2\tilde{z}\tilde{f}_2..\tilde{f}_m \neq \tilde{e}_m..\tilde{e}_2\tilde{f}_2..\tilde{f}_m$, folglich $\mathcal{T}(\mathcal{A}) \notin \mathbf{W}_{m,m}$.

IS: Gegeben sei: $\mathcal{A}$ enthält $M_{W_{m,i+1}} \Rightarrow \mathcal{T}(\mathcal{A}) \notin \mathbf{W}_{m,i+1} \wedge$ für $M_{W_{m,i+1}}$ gilt $v_s \xrightarrow{\ell} v_l \wedge v_s \xrightarrow{r} v_r$ mit $\mathbf{W}_{m,i+1} = [\![\, \ell = r \,]\!]$.
Wenn wir nun in $M_{W_{m,i+1}}$ alle Kanten mit Beschriftung e_{i+1} durch $M_{e_{i+1}}$ und diese mit Beschriftung f_{i+1} durch $M_{f_{i+1}}$ ersetzen würden erhalten wir $M_{W_{m,i}}$. Betrachten wir den Pfad ℓ, der in $M_{W_{m,i+1}}$ von v_s nach v_ℓ geführt hat. Falls ein Teilpfad e_{i+1} von einem Knoten v_e zu ein anderen Knoten v_f geführt hat, d.h. es gab eine Kantenbeschriftung $b(v_e, v_f) = e_{i+1}$, so wurde dort das Muster $M_{e_{i+1}}$ eingesetzt, d.h. an dieser Stelle sieht $M_{W_{m,i}}$ so aus:



Nun ist klar, dass wir statt dem Teilpfad e_{i+1} im alten Muster nun den Teilpfad $(e_i..e_2 z f_2..f_i x_i)^\omega$ nehmen können um von v_a nach v_f zu gelangen. Für den Fall $i = 1$ gibt es natürlich der Teilpfad $(zx_1)^\omega$. Analog gilt für einen früheren Teilpfad f_{i+1}, dass wir diesen nun durch $y_i(e_i..e_2 z f_2..f_i y_i)^{\omega-1} e_i..e_2 z f_2..f_i = (y_i e_i..e_2 z f_2..f_i)^\omega$ ablaufen können:



Da dies für alle Teilpfade e_{i+1} und f_{i+1} gilt, können wir im Muster $M_{m,i}$ nun mit $\ell' = \ell[e_{i+1}/(e_i..e_2 z f_2..f_i x_m)^\omega, f_{i+1}/(y_m e_i..e_2 z f_2..f_i)^\omega]$ von v_s nach v_ℓ und mit $r' = r[e_{i+1}/(e_i..e_2 z f_2..f_i x_m)^\omega, f_{i+1}/(y_m e_i..e_2 z f_2..f_i)^\omega]$ von v_s nach v_r gelangen. Folglich gibt es Belegungen der Variablen für die ω-Terme ℓ' und r' die in unterschiedliche Zustände überführen und damit ℓ', r' in unterschiedliche Elemente von $\mathcal{T}(\mathcal{A})$ auswerten. Da $\mathbf{W}_{m,i}$ genau $[\![\, \ell' \neq r' \,]\!]$ entspricht, ist $\mathcal{T}(\mathcal{A}) \notin W_{m,i}$. □

Aus Lemma 3.5.2 und Lemma 3.5.3 folgt nun, dass ein Minimalautomat genau dann das Muster $M_{W_{m,1}}$ enthält, wenn sein syntaktisches Monoid nicht in der Varietät $\mathbf{W}_m$ ist.
Da der Äquivalenzbeweis zwischen den Mustern $M_{R_{m,1}}$ bzw. $M_{L_{m,1}}$ und den Varietäten $\mathbf{R}_m$ bzw. $\mathbf{L}_m$ nahezu gleich verläuft, gilt folgendes Theorem.

Theorem 3.5.1. *A enthält* $M_{W_{m,1}}(M_{R_{m,1}}, M_{L_{m,1}}) \Leftrightarrow \mathcal{T}(\mathcal{A}) \notin \boldsymbol{W}_m$(*bzw.* $\boldsymbol{R}_m, \boldsymbol{L}_m$)

4 Prüfen auf Muster

Unsere Aufgabenstellung ist, anhand eines gegebenen Automaten $\mathcal{A}$ festzustellen, ob dessen syntaktisches Monoid $\mathcal{T}(\mathcal{A})$ in einer gegebenen Varietät der Trotter-Weil-Hierarchie liegt.
Es bestünde nun die Möglichkeit $\mathcal{T}(\mathcal{A})$ vollständig auszurechnen und dann die entsprechende ω-Gleichung für alle möglichen Belegungen der Variablen mit Elementen von $\mathcal{T}(\mathcal{A})$ zu überprüfen. Bezeichne nun n die Anzahl der Zustände des gegebenen Automaten $\mathcal{A}$. Schon die Größe von $\mathcal{T}(\mathcal{A})$ eines Automaten liegt jedoch im schlechtesten Fall schon in der Größenordnung n^n. Die Berechnung von $\mathcal{T}(\mathcal{A})$ stellt also bereits eine zeitliche untere Schranke von $\Omega(n^n)$ für diesen Ansatz dar.
Wir werden nun stattdessen den gegebenen Automaten $\mathcal{A}$ auf das Vorkommen von unseren Verbotsmustern aus Kapitel 3 prüfen. Hier werden wir nicht das komplette syntaktische Monoid ausrechnen müssen. Wir führen dies hier anhand der *Join*-Level $\mathbf{W}_m$ und ihrer entsprechenden Muster $M_{W_{m,1}}$ durch. Mit kleinen Modifikationen kann dies auch mit $\mathbf{R}_m$ bzw. $\mathbf{L}_m$ und den Mustern $M_{R_{m,1}}$ bzw. $M_{L_{m,1}}$ geschehen. Für die mit der FO^2-Logik in Zusammenhang stehenden Schnittlevel $\mathbf{R}_m \cap \mathbf{L}_m$ muss einmal auf $\mathbf{R}_m$ und einmal auf $\mathbf{L}_m$ getestet werden. Falls in beiden Fällen das jeweilige Muster nicht enthalten ist, so ist $\mathcal{T}(\mathcal{A})$ in $\mathbf{R}_m \cap \mathbf{L}_m$. Offensichtlich ist der doppelte Test aus komplexitätstheoretischer Sicht kein Mehraufwand. Für die Varietät **DA** können wir den Automaten selbstverständlich auch auf ihr entsprechendes Muster M_{DA} prüfen.
Wir gehen im Folgenden immer von einem gegeben Minimalautomaten aus, da wir auf diesen die Äquivalenz zwischen Muster und ω-Gleichung bewiesen haben (Theorem 3.5.1). Falls kein Minimalautomat gegeben ist, können wir den gegebenen Automaten nichtdeterministisch in logarithmisch beschränkten Raum (**NL**) [CH92] minimieren. Da die Klasse **NL** unter Komposition abgeschlossen ist, können wir danach unserem Algorithmus mit Komplexität **NL** anwenden ohne aus dieser Komplexitätsklasse herauszufallen.
Die Idee ist nun mit einer nichtdetermistischen, im Platz logarithmisch beschränkten Turingsmaschine die Abbildung des Musters auf den Automaten zu raten und zu überprüfen. Da nach dem Satz von Immerman und Szelepcsényi $\mathbf{NL} = \mathbf{co\text{-}NL}$ gilt, können wir auch die Entscheidung ob ein Muster vermieden wird in **NL** treffen. Nach Theorem 3.5.1 folgt direkt, dass wir $\mathcal{T}(\mathcal{A}) \in \mathbf{W}_m$ in **NL** entscheiden können.
Da wir außerdem jeden Algorithmus der Klasse **NL** deterministisch in polynomieller Zeit (**P**) simulieren können ($\mathbf{NL} \subseteq \mathbf{P}$), erhalten wir einen Polynomialzeitalgorithmus, der deutlich performanter als der obige Ansatz mit $\Omega(n^n)$.

Konkret muss geprüft werden, ob das Muster $M_{W_{m,1}}$ im Minimalautomaten $\mathcal{A}$ vorkommt. Dazu betrachten wir nun den folgenden Algorithmus in Pseudocode:

Algorithmus 4.1: Mustererkennung in **NL**

```
1  forall Knoten v aus M_{W_{m,1}}
2          rate Zustand p ∈ P von A
3          h(v) := p
4  if (h(v_ℓ) = h(v_r))
5          exit
6
7  forall Kantenbeschriftungen u ∈ {x_1..x_m, z, y_1..y_m}
8          j:=0
9          forall Kanten (v_1, v_2) aus M_{m,1} mit u = b(v_1, v_2)
10                 p_{1,j} := h(v_1)
11                 p_{2,j} := h(v_2)
12                 j++
13
14         while not (∀j : p_{1,j} = p_{2,j})
15                 rate a ∈ Σ
16                 p_{1,j} := δ(p_{1,j}, a)
17         forall j
18                 free(p_{1,j})
19                 free(p_{1,j})
```

Zunächst wird in den Zeilen 1-5 die Abbildung der Knoten des Musters $M_{W_{m,0}}$ auf die Zustände von $\mathcal{A}$ (Abbildung h) geraten und dabei sichergestellt, dass die Knoten v_l und v_r auf verschiedene Zustände abgebildet werden.
Die For-Schleife aus Zeile 7 iteriert über alle Kantenbeschriftungen und die For-Schleife aus Zeile 9 iteriert über alle Kanten mit Kantenbeschriftung u. Nun muss überprüft werden ob diese Kanten stimmen, d.h. es muss eine Abbildung g geben, sodass für alle Kanten (v_1, v_2) mit $u = b(v_1, v_2)$ die Bedingung $h(v_1)g(u) = h(v_2)$ erfüllt ist. Der Algorithmus prüft dies, indem er schrittweise ein Wort rät, das die geratenen Quellzustände $h(v_1)$ in die geratenen Zielzustände $h(v_2)$ überführt. Dies muss für alle Kanten mit dieser Beschriftung gleichzeitig passieren, denn es ist dem Algorithmus nicht möglich sich das gesamte Wort zu speichern. Jedoch wird dieses nicht benötigt, sondern es muss nur seine Existenz gewährleistet werden. Dazu wird getestet, ob für alle j Kanten mit Beschriftung u, die Zustände $p_{1,j}$ sich mit dem gleichen Wort in die Zustände $p_{1,j}$ überführen lassen (Zeile 14-16). Der Zustand $p_{1,j}$ wird außerdem benutzt um die Position auf dem Weg zu $p_{2,j}$ zu speichern (Zeile 16). Falls es ein solches Wort gibt, so überführt dessen Restklasse, ein Element aus $\mathcal{T}(\mathcal{A})$, ebenfalls die Zustände wie gefordert und ist deswegen ein Bildelement für $g(u)$. Schließlich werden in den Zeilen 17-19 die nicht mehr benötigten Speicherplätze für einen weiteren Schleifendurchlauf mit einer anderen Kantenbeschriftung freigegeben.
Falls also das Muster enthalten ist, so kann der Algorithmus die Abbildung h richtig raten und erkennen dass es eine passende Abbildung für g gibt und wird terminieren. Falls es keine passenden Abbildungen gibt, besteht keine Möglichkeit die Zustände so zu raten, dass es

geeignete überführende Elemente aus $\mathcal{T}(\mathcal{A})$ gibt. Damit gibt es immer für mindestens eine Kantenbeschriftung keine passende Restklasse von Σ^*, also auch kein überführendes Wort. Der Algorithmus müsste zwar in diesem Fall nicht terminieren, aber dies ist mit der Definition von **NL** vereinbar. Da außerdem nur logarithmisch viel Speicherplatz gebraucht wird um den Namen eines Zustandes zu speichern und aufgrund der festen Größe des Musters nur konstant viele Zustände gespeichert werden müssen, wird insgesamt nur logarithmischer Platz benötigt. Für einen ähnlichen, aber einfacheren Verbotsmusteralgorithmus in **NL** s. [ING13], S.7.

Damit akzeptiert der Algorithmus alle Minimalautomaten, die das Muster $M_{W_{m,1}}$ enthalten. Wegen obiger Überlegungen mit **NL** $=$ **co-NL** und Theorem 3.5.1 gilt folgendes Theorem:

Theorem 4.0.2. *Bei gegebenem Automaten $\mathcal{A}$ lässt sich in $\boldsymbol{NL}(n)$ feststellen, ob sein syntaktisches Monoid $\mathcal{T}(\mathcal{A})$ in $\boldsymbol{W}_m(\boldsymbol{R}_m, \boldsymbol{L}_m$ bzw. $\boldsymbol{DA})$ liegt.*

Wegen **NL** $\subseteq$ **P** erhalten wir außerdem einen konkreten Polynomialzeitalgorithmus für unsere Problemstellung:

Theorem 4.0.3. *Bei gegebenem Automaten $\mathcal{A}$ lässt sich in $\boldsymbol{P}(n)$ feststellen, ob sein syntaktisches Monoid $\mathcal{T}(\mathcal{A})$ in $\boldsymbol{W}_m(\boldsymbol{R}_m, \boldsymbol{L}_m$ bzw. $\boldsymbol{DA})$ liegt.*

5 Zusammenfassung und Ausblick

In Kapitel 3 wurde gezeigt, wie für die Varietät **DA** und beliebige Varietäten der Trotter-Weil Hierarchie entsprechende Verbotsmuster konstruiert werden können. Dann wurde gezeigt, dass diese genau dann in einem deterministischen endlichen Minimalautomaten vorkommen, wenn dessen syntaktisches Monoid nicht in der entsprechenden Varietät der Trotter-Weil-Hierarchie ist. In Kapitel 4 wurde mithilfe dieser Muster ein **NL**-Algorithmus konstruiert, der für einen beliebigen gegebenen DEA die Zugehörigkeit seines syntaktischen Monoids zu einer gegebenen Varietät entscheiden kann. Wegen $\mathbf{NL} \subseteq \mathbf{P}$ existiert deswegen auch ein entsprechender deterministischer Polynomialzeitalgorithmus.

Ausblick

Um den Beweis und den rekursiven Aufbau des Musters evtl. zu vereinfachen, gäbe es die Möglichkeit von einer alternativen Rekursionsvorschrift für die ω-Gleichungen (s. Kapitel 2) auszugehen, anstatt von der hier verwendeten (s. Kapitel 3).
Auch wenn im Rahmen dieser Bachelorarbeit die **NL**-Härte des Problems nicht gezeigt werden konnte, so scheint es doch nahezu unvorstellbar mit weniger als logarithmischem Platz den Automaten ausreichend untersuchen zu können. Denn es wäre für die Maschine nun nicht mehr möglich sich auch nur einen Zustand des Automaten zu speichern.
Zwar haben wir gezeigt, dass unser Problem in **P** entscheidbar ist, jedoch wäre überdies interessant zu wissen, welchen Grad das Laufzeitpolynom abhängig von der zu untersuchenden Varietät hat. Vergleichbare Untersuchungen zu Verbotsmuster [ING13] legen nahe, dass die Anzahl der zu speichernden Zustände in **NL** dem Grad des Laufzeitpolynoms entspricht. Vermutlich wird der Grad des Polynoms für die entsprechenden Level der Trotter-Weil Hierarchie exponentiell zunehmen, da die Knoten unserer Muster mit dem Level exponentiell wachsen.
Für eine konkrete Implementierung unseres Polynomialzeitalgorithmus fällt außerdem auf, dass unsere Muster „nichtdeterministisch“ sein können, d.h. es kann Knoten mit zwei Kanten geben, deren Kantenbeschriftungen identisch sind und damit müssen die nachfolgenden Knoten auch immer auf die gleichen Zustände abgebildet werden. Eine einfache Optimierung bestünde darin, diese beiden Knoten zu verschmelzen, welche nach obiger Überlegung den Grad des Laufzeitpolynoms um eins senken würde.
Schließlich muss das syntaktische Monoid eines Automaten in einer minimalen Varietät der Trotter-Weil-Hierarchie liegen, falls es in **DA** ist. Obwohl wir die Entscheidung bezüglich **DA** und für jede dieser Varietäten in **NL** bzw. **P** fällen können, bleibt offen, ob wir ähnlich effizient diese minimale Varietät bestimmen können. Leider hängt dabei die minimale Varietät von dem gegebenem Automaten ab und damit kann auch nicht mehr die Größe der zu untersuchenden Muster als konstant angenommen werden.

Literaturverzeichnis

[Bir70] A. Biryukov. Varieties of idempotent semigroups. *Algebra and Logic*, 9(3):153–164, 1970. (Zitiert auf Seite 15)

[CH92] S. Cho, D. T. Huynh. The parallel complexity of finite-state automata problems. *Inf. Comput.*, 97(1):1–22, 1992. (Zitiert auf Seite 35)

[DGK08] V. Diekert, P. Gastin, M. Kufleitner. A survey on small fragments of first-order logic over finite words. *International Journal of Foundations of Computer Science*, 19(03):513–548, 2008. (Zitiert auf Seite 15)

[Fen71] C. Fennemore. All varieties of bands I, II. *Mathematische Nachrichten*, 48:237–252, 253–262, 1971. (Zitiert auf Seite 15)

[Ger70] J. Gerhard. The lattice of equational classes of idempotent semigroups. *Journal of Algebra*, 15(2):195 – 224, 1970. (Zitiert auf Seite 15)

[HMU07] J. E. Hopcroft, R. Motwani, J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Pearson Addison-Wesley, Upper Saddle River, NJ, 3 Auflage, 2007. (Zitiert auf Seite 10)

[ING13] S. Iván, J. Nagy-György. On the structure and syntactic complexity of generalized definite languages. *CoRR*, abs/1304.5714, 2013. (Zitiert auf den Seiten 37 und 39)

[KL12] M. Kufleitner, A. Lauser. The Join Levels of the Trotter-Weil Hierarchy Are Decidable. In *Mathematical Foundations of Computer Science 2012*, Band 7464 von *Lecture Notes in Computer Science*, S. 603–614. Springer Berlin Heidelberg, 2012. (Zitiert auf den Seiten 12 und 14)

[KW10] M. Kufleitner, P. Weil. On the lattice of sub-pseudovarieties of DA. *Semigroup Forum*, 81(2):243–254, 2010. (Zitiert auf Seite 15)

[KW12a] M. Kufleitner, P. Weil. The FO^2 alternation hierarchy is decidable. In *CSL*, Band 16 von *LIPIcs*, S. 426–439. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012. (Zitiert auf den Seiten 7 und 15)

[KW12b] M. Kufleitner, P. Weil. On logical hierarchies within FO^2-definable languages. *Logical Methods in Computer Science*, 8(3), 2012. (Zitiert auf Seite 15)

[Lau09] A. Lauser. *Fragmente einer Intervall-Logik*. Diplomarbeit nr.2823, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, 2009. (Zitiert auf Seite 10)

[LPS08] K. Lodaya, P. Pandya, S. Shah. Marking the chops: an unambiguous temporal logic. In G. Ausiello, J. Karhumäki, G. Mauri, L. Ong, Herausgeber, *Fifth Ifip International Conference On Theoretical Computer Science – Tcs 2008*, Band 273 von *IFIP International Federation for Information Processing*, S. 461–476. Springer US, 2008. (Zitiert auf Seite 15)

[TW97] P. Trotter, P. Weil. The lattice of pseudovarieties of idempotent semigroups and a non-regular analogue. *algebra universalis*, 37(4):491–526, 1997. (Zitiert auf den Seiten 7 und 15)

[TW98] 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*, STOC '98, S. 234–240. ACM, New York, NY, USA, 1998. (Zitiert auf Seite 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