Institut für Formale Methoden der Informatik
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3337

Minimierung von Automaten mit einer beschränkten Anzahl von Fehlern

Franz G. Jahn

Studiengang:	Informatik
Prüfer:	Prof. Dr. Volker Diekert
Betreuer:	Dr. Manfred Kufleitner
begonnen am:	22. März 2012
beendet am:	21. September 2012
CR-Klassifikation:	F.1.1; F.1.2; F.4.3

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Algorithmen

1 Einleitung

Benedikt, Puppis und Riveros haben 2011 gezeigt, dass es für eine reguläre Sprachen L und eine andere reguläre Sprache K entscheidbar ist, ob sich L so in K einbetten lässt, dass man auf jedes Wort aus L nur eine beschränkte Anzahl von Änderungsoperationen (ein Zeichen einfügen, löschen oder austauschen) anwenden muss, um ein Wort der Sprache K zu erhalten [BPR11]. Hierbei muss es eine maximale Anzahl der Änderungsoperationen geben, die für kein Wort der Sprache L überschritten wird. Die Idee dieser Problemstellung basiert auf einer Eingabemenge L, die einer Spezifikation K zunächst nicht genügt. Es stellt sich dann die Frage, ob eine Möglichkeit existiert, die Eingabe L in einem Reparaturvorgang so zu modifizieren, dass die Spezifikation anschließend erfüllt wird. Es werden hierbei zwei Reparaturmodi unterschieden: „Streaming“ und „Nonstreaming“.

Die Intuition dieser beiden Modi ist dabei, dass im „Nonstreaming“ Modus das ganze Wort betrachtet wird und darauf basierend Änderungsoperationen vorgenommen werden, während im „Streaming“ Modus das Wort zeichenweise verarbeitet wird und für die Entscheidung, welche Änderungsoperationen durchgeführt werden, nur der bis dahin gelesene Teil des Wortes betrachtet wird. Die Komplexität der Entscheidung, ob die Reparaturdistanz endlich ist, hängt dabei sowohl vom gewählten Reparaturmodus, als auch von der Repräsentation der Sprache ab. Für die Repräsentation als Automaten, kann zur Entscheidung dieses Problems zum einen der Graph der starken Zusammenhangskomponenten, also die Struktur des Automaten, und zum anderen ein aus den Automaten konstruiertes Erreichbarkeitsspiel betrachtet werden.

Betrachtet man eine endliche Distanz zwischen zwei Sprachen als Relation, so ergibt sich auf natürliche Weise ein Äquivalenzbegriff für Sprachen. Zwei Sprachen L und K sind hierbei äquivalent, wenn sie sich gegenseitig mit beschränkt vielen Fehlern ineinander einbetten lassen. Im Rahmen dieser Arbeit, wird der Äquivalenzbegriff für die beiden Reparaturmodi „Streaming“ und „Nonstreaming“ genauer untersucht. Das Hauptaugenmerk liegt dabei auf der Frage, ob und wie sich für die Äquivalenzklassen minimale Automaten konstruieren lassen, die jeweils eine Sprache als Repräsentant dieser Klasse erkennen.

Als Basis für die Minimierung untersuchen wir in Kapitel 4 einige Sprach- und Automatenklassen, die in diesem Zusammenhang interessante Eigenschaften aufweisen. Hierbei betrachten wir sowohl die Charakterisierung der Sprachklassen über Automatenmodelle, als auch algebraische und verbandstheoretische Charakterisierungen dieser Sprachklassen. Insbesondere präsentieren wir Automatenmodelle für diejenigen Sprachklassen, deren algebraische Entsprechung Ideale, Filter und deren Boolesche Kombinationen sind. Schließlich betrachten wir noch für einige spezielle Automatenklassen das Inklusionsproblem.

Mit Hilfe dieser Ergebnisse, zeigen wir in Kapitel 5 zunächst, dass es für den Reparaturmodus „Nonstreaming“ PSPACE hart ist zu entscheiden, ob die Anzahl der starken Zusammenhangskomponenten für einen (nichtdeterministischen) Automaten minimal ist. Für den Modus „Streaming“ präsentieren wir anschließend einen Polynomialzeitalgorithmus zur Minimierung eines deterministischen Automaten innerhalb einer Äquivalenzklasse. Hierbei bedienen wir uns eines Erreichbarkeitsspiels, welches zu einem Automaten konstruiert werden kann. Die Konstruktion des minimalen Automaten sorgt dabei zunächst nur dafür, dass das daraus konstruierte Spiel spezielle Eigenschaften erfüllt. Wir zeigen, dass daraus die Minimalität folgt.

2 Grundlagen

In diesem Kapitel werden die grundlegenden Notationen und Begrifflichkeiten eingeführt. In Abschnitt 2.1 werden zunächst Automaten und deren Eigenschaften beschrieben, die wir in dieser Arbeit benötigen, in Abschnitt 2.2 werden Abbildungen zwischen Sprachen in Form von Prozessoren und deren Realisierung in Form von Transducer, einer speziellen Art von Automaten, beschrieben. Erreichbarkeitsspiele werden in Abschnitt 2.3 eingeführt. In Abschnitt 2.4 führen wir, basierend auf der Levenshtein-Distanz, verschiedene Möglichkeiten ein, eine Distanz zwischen zwei Sprachen zu definieren.

In dieser Arbeit bezeichnet Σ immer ein endliches Alphabet. Die Menge der endlichen Wörter über dem Alphabet Σ bezeichnen wir mit Σ^*. Für das leere Wort schreiben wir ε. Mit $w \in \Sigma^*$ bezeichnen wir ein endliches Wort über dem Alphabet Σ, mit $|w|$ bezeichnen wir die Länge von w und mit $w[i]$ bezeichnen wir für $1 \leq i \leq |w|$ das Zeichen an der i-ten Stelle des Wortes w.

2.1 Automaten

Wir beschreiben einen endlichen Automaten als 5-Tupel $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$. Hierbei bezeichnet Q eine endliche Menge von Zuständen, Σ ein endliches Alphabet, $\delta \subseteq Q \times \Sigma \times Q$ die Übergangsrelation, $Q_0 \subseteq Q$ eine Menge von Startzuständen und $F \subseteq Q$ eine Menge von Endzuständen. Für einen Übergang $(p, a, q) \in \delta$ schreiben wir auch $p \xrightarrow{a} q$. Die Übergangsrelation erweitern wir induktiv, wie folgt, auf Wörter: $q \xrightarrow{\varepsilon} q$ für alle $q \in Q$; und zudem $p \xrightarrow{au} q$ wenn es einen Zustand $r \in Q$ gibt, sodass $p \xrightarrow{a} r$ und $r \xrightarrow{u} q$. Wir gehen davon aus, dass alle Zustände erreichbar sind, *d.h.* für jeden Zustand $q \in Q$ existieren $q_0 \in Q_0$ und $w \in \Sigma^*$ so, dass $q_0 \xrightarrow{w} q$ gilt.

Die Übergangsrelation eines Automaten ist deterministisch, wenn für alle Zustände $p \in Q$ und alle Zeichen $a \in \Sigma$ höchstens ein Zustand $q \in Q$ mit $p \xrightarrow{a} q$ existiert. Ein Automat ist deterministisch, wenn seine Übergangsrelation deterministisch ist und $|Q_0| = 1$. Wir bezeichnen einen Automaten als *vollständig*, wenn es für alle $p \in Q$ und $a \in \Sigma$ mindestens einen Zustand $q \in Q$ gibt mit $p \xrightarrow{a} q$. Ein Automat ist *co-erreichbar*, wenn von jedem Zustand aus ein Endzustand erreichbar ist, d.h. für alle $p \in Q$ existiert ein $w \in \Sigma^*$ sodass $p \xrightarrow{w} q$ für ein $q \in F$. Da wir davon ausgehen, dass alle Zustände erreichbar sind, ist ein Automat *trim*, wenn er co-erreichbar ist.

Ein Lauf eines Automaten $\mathcal{A}$ auf einem Wort $w = w[1]w[2]\dots w[n]$ ist eine Folge von Zuständen $q_0 q_1 \dots q_n$ für die $q_0 \in Q_0$ und $q_{i-1} \xrightarrow{w[i]} q_i$ für $1 \leq i \leq n$ gilt. Einen Lauf

bezeichnen wir als akzeptierend, wenn $q_n \in F$. Ein Wort $w \in \Sigma^*$ wird von $\mathcal{A}$ akzeptiert, wenn es einen akzeptierenden Lauf von $\mathcal{A}$ auf w gibt. Die Sprache der von $\mathcal{A}$ akzeptierten Wörter bezeichnen wir mit $L(\mathcal{A}) = \{w \in \Sigma^* \mid w \text{ wird von } \mathcal{A} \text{ akzeptiert}\}$.

Wir betrachten Automaten zudem als gerichtete gelabelte Graphen. Die Knotenmenge des Graphen entspricht dabei der Zustandsmenge des Automaten und für jeden Übergang $p \xrightarrow{a} q$ besitzt der Graph eine mit a beschriftete Kante von p nach q.

Für einen Zustand $q \in Q$ bezeichnen wir mit $\mathcal{C}(q)$ die starke Zusammenhangskomponente der wechselseitig von q aus erreichbaren Zustände. Wir bezeichnen eine Zusammenhangskomponente C als *final*, wenn von C aus ein Endzustand erreicht werden kann, d.h. wenn ein Zustand $p \in C$ und ein Wort $w \in \Sigma^*$ existieren mit $p \xrightarrow{w} q$ für ein $q \in F$. Für einen endlichen Automaten $\mathcal{A}$ bezeichnen wir mit $SCC(\mathcal{A})$ die Menge der starken Zusammenhangskomponenten von $\mathcal{A}$.

Wir bezeichnen einen Automaten $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ als stark zusammenhängend, wenn er aus einer einzigen starken Zusammenhangskomponente besteht, *d.h.* wenn für alle $p, q \in Q$ ein Wort $w \in \Sigma^*$ existiert mit $p \xrightarrow{w} q$.

Für einen Automaten $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ und eine Zustandsmenge $D \subseteq Q$ bezeichnen wir mit $\mathcal{A}|D$ den Automaten, welchen wir aus $\mathcal{A}$ erhalten, wenn wir $\mathcal{A}$ auf die Zustände aus D einschränken und alle Zustände aus D zu Start- und Endzuständen machen, d.h. $\mathcal{A}|D = (D, \Sigma, \delta', D, D)$ mit $\delta' = \delta \cap (D \times \Sigma \times D)$.

2.2 Prozessor und Transducer

Ein *Prozessor* ist eine Abbildung $p : \Sigma \to \Gamma$, welche Wörter über einem endlichen Alphabet Σ auf Wörter über einem endlichen Alphabet Γ abbildet.

Ein *Transducer* oder genauer ein sequenzieller Zeichen zu Wort Transducer ist ein spezieller deterministischer Automat, welcher ein Eingabewort v zeichenweise verarbeitet und dabei ein Ausgabewort w erzeugt. Die Ausgabe hängt dabei jeweils vom aktuellen Zeichen und vom Zustand ab, in welchem sich der Transducer befindet.

Wir beschreiben einen Transducer als 6-Tupel $\mathcal{T} = (Q, \Sigma, \Gamma, \delta, q_0, \Omega)$. Wie bei einem deterministischen endlichen Automaten bezeichnet Q eine endliche Menge von Zuständen, Σ ein endliches Eingabealphabet und q_0 den Startzustand. Γ bezeichnet ein endliches Ausgabealphabet und δ bezeichnet die Übergangsfunktion $\delta : Q \times \Sigma \to \Gamma^* \times Q$. Wir schreiben auch $p \xrightarrow{a/w} q$ für $\delta : (p, a) \mapsto (w, q)$. Mit Ω bezeichnen wir eine finale Ausgabefunktion $\Omega : Q \to \Gamma^*$, welches nach Abschluss der Lesevorgangs noch eine Ausgabe erzeugen kann.

Ein Lauf von $\mathcal{T}$ auf einem Eingabewort $v = a_1 a_2 \dots a_n, a_i \in \Sigma$ ist von der Form $q_0 \xrightarrow{a_1/w_1} q_1 \xrightarrow{a_2/w_2} q_2 \dots q_{n-1} \xrightarrow{a_n/w_n} q_n \xrightarrow{\varepsilon/w_{n+1}}$, hierbei ist $w_{n+1} = \Omega(q_n)$ und $w_i \in \Gamma^*$. Das Ausgabewort ist $w = w_1 w_2 \dots w_n w_{n+1}$. Ein Transducer $\mathcal{T}$ definiert somit eine Abbildung von Σ^* nach Γ^*. Wir schreiben daher auch $\mathcal{T}(v) = w$.

Bemerkung 2.1. Ein Transducer realisiert einen Prozessor, aber nicht jeder Prozessor kann durch einen Transducer realisiert werden.

Ein Prozessor kann beispielsweise für $n \in \mathbb{N}$ alle Wörter der Form $a^n b$ auf b^n abbilden, aber alle Wörter der Form $a^n c$ auf c^n. Ein Transducer könnte in diesem Fall erst nach lesen des letzten Zeichens der Eingabe das erste Zeichen der Ausgabe schreiben. Da die Ausgabe eines Transducers aber nur vom aktuell gelesenen Zeichen sowie dem aktuellen Zustand abhängt, kann er beim Lesen des letzten Zeichens keine beliebig lange Ausgabe mehr erzeugen. Daher lässt sich eine Abbildung dieser Form nicht durch einen Transducer realisieren.

2.3 Erreichbarkeitsspiele

Ein *Erreichbarkeitsspiel* wird von zwei Spielern (*Adam* und *Eva*) auf einer endlichen Spielarena bestritten. Eine *Spielarena* ist ein gefärbter gerichteter Graph $\Gamma = (V, E, \rho)$, wobei V eine Menge von Knoten, $E \subseteq V \times V$ eine Menge von Kanten und $\rho : V \to \{Adam, Eva\}$ eine Knotenfärbung des Graphen ist, die jedem Knoten einen der beiden Spieler zuordnet. Für einen Knoten $v \in V$ bezeichnen wir mit $succ(v) = \{v' \in V \mid (v, v') \in E\}$ die Menge aller direkten Nachfolgerknoten. Für Adam und Eva bezeichnen wir mit $V_A = \{v \in V \mid \rho(v) = Adam\}$ bzw. $V_E = \{v \in V \mid \rho(v) = Eva\}$ die Menge der Knoten, bei welchen Adam bzw. Eva am Zug sind. Wir sprechen daher auch von *Adams Knoten* und *Evas Knoten*. Ein Spieler zieht, indem er zu der aktuellen Spielposition $v \in V$ aus den direkten Nachfolgerknoten $succ(v)$ eine neue Spielposition $v' \in succ(v)$ auswählt.

Ein *Spiel* s in der Spielarena Γ ist ein maximaler Pfad in Γ, *d.h.* s ist entweder ein unendlicher Pfad (ein *unendliches Spiel*) $s = v_0 \to v_1 \to v_2 \to \ldots$ oder ein endlicher Pfad (ein *endliches Spiel*) $s = v_0 \to v_1 \to \ldots \to v_n$ mit $v_i \in V$ und $succ(v_n) = \emptyset$. *Adam* (bzw. *Eva*) gewinnt ein Spiel s genau dann, wenn $s = v_0 \to v_1 \to \ldots \to v_n$ endlich ist und $\rho(v_n) = Adam$ (bzw. $\rho(v_n) = Eva$).

Wir bezeichnen einen Knoten v als *final*, wenn er keine ausgehende Kanten besitzt, *d.h.* $succ(v) = \emptyset$. Die Menge der finalen Knoten einer Spielearena $\Gamma = (V, E, \rho)$ bezeichnen wir mit $V_{fin} = \{v \in V \mid succ(v) = \emptyset\}$.

Eine *Strategie* für Adam ist eine Abbildung $\tau_A : V_A \setminus V_{fin} \to V$ welche jedem Knoten $v \in V_A \setminus V_{fin}$ einen Knoten $v' \in succ(v)$ zuordnet. Ein endliches Spiel $s = v_0 ... v_n$ ist mit der einer Strategie τ_A *konform*, wenn für alle Knoten v_i mit $0 \leq i < n$ aus $\rho(v_i) = Adam$ folgt, dass $v_{i+1} = \tau_A(v_i)$. Für ein unendliches Spiel s muss diese Bedingung für alle $0 \leq i$ gelten, damit s mit τ_A konform ist. Analog definieren wir eine Strategie τ_E für Eva. Eine Strategie τ_x wird für einen Spieler $x \in \{Adam, Eva\}$ an einer Position v als *Gewinnstrategie* bezeichnet, wenn Spieler x jedes bei v startende und mit τ_x konforme Spiel gewinnt.

Bemerkung 2.2. Das Wort *Spiel* kann in der deutschen Sprache sowohl für das Spiel im Sinne des Regelwerkes zu einer Spieldurchführung stehen, als auch für den Ablauf eines Spiels stehen. Für eine bessere Verständlichkeit sprechen wir im Folgenden von *Erreichbarkeitsspiel* wenn wir das Regelwerk meinen und bezeichnen mit *Spiel* immer den Ablauf oder die Durchführung eines Spiels.

2.4 Distanz regulärer Sprachen

Um den Begriff der Distanz innerhalb und zwischen Sprachen zu verstehen, betrachten wir zunächst die Distanz zwischen einzelnen Worten. Basis für die verschiedenen Definitionen ist die Levenshtein-Distanz. Ausgehend davon, können verschiedene Betrachtungen durchgeführt werden, auf Grund derer man sowohl eine innere Distanz (als die Distanz zwischen den zu einer Sprache gehörenden Worten), als auch die Distanz zwischen zwei verschiedenen Sprachen auf verschiedene Weisen definieren kann.

Die *Levenshtein-Distanz* zwischen zwei Wörtern v und w ist minimale Anzahl atomarer Operationen (Einfügen, Löschen oder Ersetzen eines einzelnen Zeichens), die notwendig sind um v in w zu überführen. Wir schreiben dafür auch $\mathrm{dist}(v, w)$.

Beispiel 2.1. Die Levenshtein-Distanz zwischen $v = abab$ und $w = baba$ ist zwei.

1. $v = abab$
2. $v' = bab$ (Löschen des ersten Zeichens)
3. $w = baba$ (Einfügen eines „a“ am Ende)

Es ist bekannt, dass es mittels dynamischer Programmierung möglich ist, die Levenshtein-Distanz von zwei gegebenen Wörtern v und w in $\mathcal{O}(|v| \cdot |w|)$ zu berechnen (siehe [SK00]).

Die Levenshtein-Distanz spielt aber auch in vielen weiteren Problemen im Bezug auf Wörter, Sprachen und Sprachklassen eine wichtige Rolle. Betrachten wir den Abstand von einem Wort v zu einer Sprache L, so können wir diesen Abstand als den kleinsten Abstand zu einem Wort in L betrachten: $\mathrm{dist}(v, L) = \min\{\mathrm{dist}(v, w) \mid w \in L\}$. Ist das Wort $w \in L$ mit dem minimalen Abstand zu v eindeutig, kann dies beispielsweise zur Fehlerkorrektur genutzt werden. Für reguläre Sprachen wurde dieses Problem von Wagner in [Wag74] betrachtet.

Betrachtet man eine einzelne Sprache L, so kann man mit der minimalen Distanz zweier Worte v und w aus L die innere Distanz einer Sprache beschreiben $\mathrm{dist}(L) = \min\{\mathrm{dist}(v, w) \mid w, v \in L \wedge w \neq v\}$. Hierbei gibt die Distanz der Sprache an, wie viele Änderungsoperationen minimal notwendig sind, um von einem Wort der Sprache zu einem anderen Wort der Sprache zu gelangen. Diese Betrachtung bietet sich beispielsweise dann an, wenn Informationen zur Übertragung codiert werden sollen und eine beschränkte Anzahl von Übertragungsfehlern erkannt werden sollen. Hierfür wird, vor allem im technischen Kontext, auch oft die Hamming-Distanz betrachtet [Ham50].

Benedikt, Puppis und Riveros [BPR11] definieren die Distanz von einer Einschränkungssprache R (*R*estriction language) zu einer Zielsprache T (*T*arget language) als die maximale Distanz eines Wortes $w \in R$ zur Sprache T: $\mathrm{dist}(R, T) = \max_{w \in R} \mathrm{dist}(w, T)$. Dieser Distanzbegriff gibt die maximale Anzahl von Änderungsoperationen an, die notwendig sind um ein Wort aus R in ein Wort aus T zu transformieren. Diese Distanz ist potentiell unendlich.

Bemerkung 2.3. Da die atomaren Änderungsoperationen (Einfügen, Löschen und Ersetzen eines einzelnen Zeichens) umkehrbar sind, ist die Levenshtein-Distanz zwischen zwei Wörtern symmetrisch: $\text{dist}(v, w) = \text{dist}(w, v)$.

Für die Distanz zwischen zwei Sprachen R und T gilt dies im Allgemeinen nicht.

Beispiel 2.2. Betrachten wir die Sprachen $L_1 = a^*$ und $L_2 = \{a, b\}^*$, so ist die Distanz $\text{dist}(L_1, L_2) = 0$, da jedes Wort aus L_1 auch zur Sprache L_2 gehört, umgekehrt ist dies nicht der Fall. Die Distanz $\text{dist}(L_2, L_1)$ ist sogar unendlich, da für jede natürliche Zahl $n \in \mathbb{N}$ ein Wort $b^{n+1} \in L_2$ existiert, für welches die Distanz $\text{dist}(b^{n+1}, L_1) = n + 1$ und somit größer als n ist.

Betrachtet man einen Transducer $\mathcal{T}$ als Abbildung $\mathcal{T} : R \to T$, so kann man für ein Eingabewort v und sein Bild $\mathcal{T}(v) = w$ die Distanz $\text{dist}(v, w)$ als Kosten der Abbildung betrachten. Für den Transducer können wir somit die Kostenfunktion $\text{cost}_{\mathcal{T}} : R \to \mathbb{N}$ mit $\text{cost}_{\mathcal{T}}(v) = \text{dist}(v, \mathcal{T}(v))$ betrachten. Bezüglich einer Eingabesprache R können wir dafür die maximalen Kosten eines Transducers $\mathcal{T}$ mit $\max_{w \in R} \text{dist}(w, \mathcal{T}(w))$ angeben.

Für eine Eingabesprache R und eine Zielsprache T bezeichnen wir mit der *Streaming Repair Distanz* die maximalen Kosten, die ein Transducer, welcher R auf die Zielsprache T abbildet, bezüglich aller Worte $w \in R$ aufweist. Die Streaming Repair Distanz gibt also an, wie viele Änderungen ein Transducer an einem Wort aus der Eingabesprache R maximal vornehmen muss, um ein Wort aus der Sprache T zu erhalten. Da der Transducer zu dem Zeitpunkt, zu dem die Änderungen vorgenommen werden, noch nicht das gesamte Wort kennt, sondern auf dem "Eingabestrom" arbeitet, bezeichnen wir den Änderungevorgang als "Streaming Repair" und die daraus resultierende Distanz zwischen Eingabe und Ausgabe als "Streaming Repair Distanz". Formal kann man diese Distanz wie folgt angeben: $\text{dist}_{stream}(R, T) = \min_{\{\mathcal{T} \mid \mathcal{T}: R \to T\}}(\max_{w \in R} \text{cost}_{\mathcal{T}}(w))$.

Die Fragestellung, ob für zwei gegebene Sprachen R und T die Distanz $\text{dist}(R, T)$ endlich ist, wird als *Bounded Repair Problem* bezeichnet. Die Frage, ob die Streaming Repair Distanz $\text{dist}_{stream}(R, T)$ endlich ist, bezeichnen wir als *Bounded Streaming Repair Problem*. Auf beide Probleme gehen wir in den folgenden Kapiteln noch genauer ein.

3 Problemstellung

Als Ausgangspunkt unserer Problemstellung dienen die Ergebnisse von Benedikt, Puppis und Riveros. Diese haben sich in [BPR11] mit dem Bounded Repair Problem und dem Bounded Streaming Repair Problem beschäftigt. Sie analysierten, wie für verschiedene Repräsentationen von regulären Einschränkungs- und Zielsprachen festgestellt werden kann, ob die Distanz (bzw. die Streaming Repair Distanz) endlich ist und betrachteten die Komplexität für die verschiedenen Repräsentationen. Als mögliche Repräsentation wählten sie deterministische endliche Automaten, nichtdeterministische endliche Automaten und Lineare temporale Logik (LTL) über endlichen Wörtern.

In diesem Kapitel betrachten wir zunächst die Ergebnisse von Benedikt, Puppis und Riveros und zeigen anschließend, welche daraus resultierenden, offenen Fragen in dieser Arbeit behandelt werden. In Abschnitt 3.1 betrachten wir zunächst ihre Ergebnisse im Hinblick auf das Bounded Repair Problem, in Abschnitt 3.2 betrachten wir die Ergebnisse zum Bounded Streaming Repair Problem. Die darauf aufbauende Problemstellung dieser Arbeit sowie Vereinfachungen der Betrachtung, die sich für diese Arbeit anbieten, werden in Abschnitt 3.3 betrachtet.

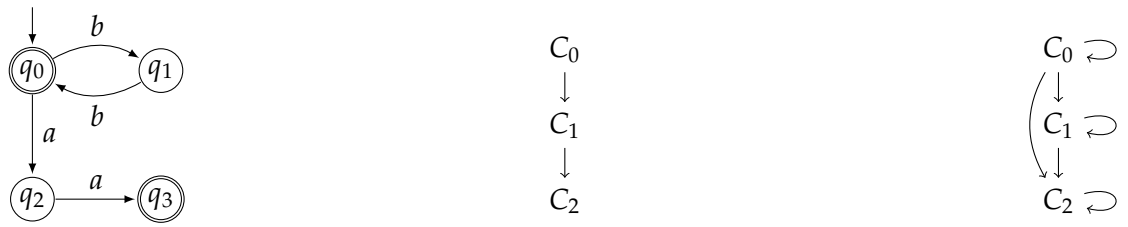
3.1 Bounded Repair

3.1.1 $Dag(\mathcal{A})$, $Dag^*(\mathcal{A})$ und Überdeckung

Für einen Automaten $\mathcal{A}$ bezeichnen wir mit $Dag(\mathcal{A})$ den gerichteten azyklischen Graphen der finalen starken Zusammenhangskomponenten von $\mathcal{A}$, *d.h.* $Dag(\mathcal{A})$ enthält die starken Zusammenhangskomponenten von $\mathcal{A}$, von welchen aus ein Endzustand erreichbar ist. Mit $Dag^*(\mathcal{A})$ bezeichnen wir den reflexiv-transitiven Abschluss von $Dag(\mathcal{A})$. Beide Graphen enthalten zunächst keine weiteren Informationen über die darin dargestellten Zusammenhangskomponenten. Um mit Hilfe dieser Graphen sinnvoll Aussagen über die zugrunde liegenden Automaten treffen zu können, verwenden wir den Begriff der Überdeckung.

Sei $\pi = C_1 \cdots C_n$ ein Pfad im gerichteten azyklischen Graphen $Dag(\mathcal{A})$ eines Automaten $\mathcal{A}$ und sei $\pi' = C'_1 \cdots C'_m$ ein Pfad in $Dag^*(\mathcal{B})$ für einen beliebigen Automaten $\mathcal{B}$. Wir sagen π' *überdeckt* π, wenn gilt $n = |\pi| = |\pi'| = m$ und wenn für alle $1 \leq i \leq n$ gilt: $L(\mathcal{A}|C_i) \subseteq L(\mathcal{B}|C'_i)$.

Beispiel 3.1. Abbildung 3.1 zeigt den Zusammenhang zwischen einem Automaten $\mathcal{A}$ und den zugehörigen Graphen $Dag(\mathcal{A})$ und $Dag^*(\mathcal{A})$.



(a) $\mathcal{A}$ (b) $Dag(\mathcal{A})$ (c) $Dag^*(\mathcal{A})$

Abbildung 3.1: Ein Automat $\mathcal{A}$ mit den zugehörigen Graphen $Dag(\mathcal{A})$ und $Dag^*(\mathcal{A})$

Links befindet sich der endliche Automat $\mathcal{A}$, in der Mitte der dazugehörige Graph der starken Zusammenhangskomponenten $Dag(\mathcal{A})$ und rechts dessen reflexiv-transitiver Abschluss $Dag^*(\mathcal{A})$.

3.1.2 Charakterisierung des Bounded Repair Problems

Als Basis für die Analyse des Bounded Repair Problems wählen Benedikt, Puppis und Riveros zunächst sowohl für die Einschränkungssprache R, als auch für die Zielsprache T die Repräsentation als nichtdeterministische endliche Automaten $\mathcal{R}$ und $\mathcal{T}$. Sie zeigen in [BPR11] Theorem 2, dass sich die das Bounded Repair Problem für diese Repräsentation mit Hilfe der gerichteten azyklischen Graphen der beiden Automaten wie folgt entscheiden lässt.

Satz 3.1 (Benedikt, Puppis und Riveros)**.** *Für zwei gegebene endliche Automaten $\mathcal{R}$ und $\mathcal{T}$ sind die folgenden Bedingungen äquivalent:*

1. *Die Distanz von $L(\mathcal{R})$ zu $L(\mathcal{T})$ ist endlich.*
2. *Jeder Pfad in $Dag(\mathcal{R})$ wird von einem Pfad in $Dag^*(\mathcal{T})$ überdeckt.*
3. $dist(L(\mathcal{R}), L(\mathcal{T})) \leq (1 + |Dag(\mathcal{R})|) \cdot |\mathcal{T}|$

Um die Idee hinter Satz 3.1 zu verstehen, bietet es sich an, zunächst eine einzelne starke Zusammenhangskomponente C_R im Automat $\mathcal{R} = (Q_R, \Sigma_R, \delta_R, Q_{0,R}, F_R)$ der Einschränkungssprache R zu betrachten. Existiert innerhalb der Zusammenhangskomponente ein Pfad $q_0 \xrightarrow{a_1} q_1 \dots q_{n-1} \xrightarrow{a_n} q_n$ (mit $q_i \in C_R$), so muss es innerhalb des Automaten $\mathcal{T} = (Q_T, \Sigma_T, \delta_T, Q_{0,T}, F_T)$ ebenfalls eine starke Zusammenhangskomponente C_T geben innerhalb welcher ein Pfad $q'_0 \xrightarrow{a_1} q'_1 \dots q'_{n-1} \xrightarrow{a_n} q'_n$ (mit $q'_i \in C_T$) existiert. Andernfalls könnte man für $\mathcal{R}$ ein Wort $w_k = t(a_1 \dots a_n u)^k v$ mit $p \xrightarrow{t} q_0$, $q_n \xrightarrow{u} q_0$ und $q_0 \xrightarrow{v} q$ für $q \in F_R$ konstruieren. Nach Konstruktion gilt für alle $k \in \mathbb{N} : w_k \in L(\mathcal{R})$. Existiert in $\mathcal{T}$ keine starke Zusammenhangskomponente C_T mit einem Pfad $q'_0 \xrightarrow{a_1} q'_1 \dots q'_{n-1} \xrightarrow{a_n} q'_n$ ($q'_i \in C_T$), so muss während eines Laufes von $\mathcal{T}$ auf dem Teilwort $a_1 \dots a_n$ entweder die Zusammenhangskomponente gewechselt werden, oder es ist eine Korrekturoperation auf dem Teilwort $a_1 \dots a_n$ notwendig, damit das Bild $f(w_k)$ von $\mathcal{T}$ akzeptiert wird. Da die Anzahl der starken

Zusammenhangskomponenten fix ist, folgt somit, dass es für jede Schranke $i \in \mathbb{N}$ ein $k \in \mathbb{N}$ existiert, sodass gilt $\text{dist}(w_k, T) > i$. Somit folgt $\text{dist}(R, T) = \infty$.

Erweitert man diese Idee von einer starken Zusammenhangskomponente auf die Betrachtung ganzer Automaten und setzt man die obige Argumentation logisch fort, so ergeben sich die Aussagen aus Satz 3.1. Uns genügt hier die Tatsache, dass die Distanz der Sprachen $L(\mathcal{R})$ zur Sprache $L(\mathcal{T})$ genau dann endlich ist, wenn jeder Pfad in $Dag(\mathcal{R})$ von einem Pfad in $Dag^*(\mathcal{T})$ überdeckt wird.

Da deterministische endliche Automaten nur eine spezielle Ausprägung von nichtdeterministischen Automaten darstellen, ist es offensichtlich, dass diese Charakterisierung auch für deterministische endliche Automaten gilt.

3.1.3 Komplexität

In Abschnitt 3.2.1 wurde mit Satz 3.1 der Zusammenhang zwischen dem Bounded Repair Problem und den Graphen der starken Zusammenhangskomponenten aufgezeigt. Darauf aufbauend lässt sich die Komplexität des Bounded Repair Problems analysieren. Benedikt, Puppis und Riveros stützen ihre Analyse auf folgenden Algorithmus, welcher für auf Eingabe eines endlichen Automaten $\mathcal{R}$, eines endlichen Automaten $\mathcal{T}$ und eins Pfades $\pi = C_1 \dots C_k$ in $Dag(\mathcal{R})$ prüft, ob dieser durch einen Pfad $C'_1 \dots C'_k$ in $Dag^*(\mathcal{T})$ überdeckt wird:

Algorithmus 3.1 Pfad Überdeckung

```
1:  function PathCoverability(R, T, π)
2:      let π = C_1 ... C_k
3:      let Dag*(T) = (V', E')
4:      F' ← ∅
5:      F ← ∅
6:      for C ∈ V' do
7:          if CheckContainment(R, T, C_1, C) then
8:              F ← F ∪ {C}
9:          end if
10:     end for
11:     for i ∈ {2 ... k} do
12:         F' ← F
13:         F ← ∅
14:         for (C' ∈ F' and (C', C) ∈ E') do
15:             if CheckContainment(R, T, C_i, C) then
16:                 F ← F ∪ {C}
17:             end if
18:         end for
19:     end for
20:     return (F ≠ ∅)
21: end function
```

Algorithmus 3.1 ist offensichtlich von polynomieller Laufzeit, relativ zu einem Orakel CheckContainment$(\mathcal{R}, \mathcal{T}, C_R, C_T)$, welches die Inklusion $L(\mathcal{T}|C_T) \subseteq L(\mathcal{R}|C_R)$ prüft. Die Komplexität des Bounded Repair Problems hängt somit zum einen wesentlich von der Komplexität des Orakels CheckContainment$(\mathcal{R}, \mathcal{T}, C_R, C_T)$ und zum anderen von der Anzahl der notwendigen Aufrufe von PathCoverability$(\mathcal{R}, \mathcal{T}, \pi)$ bzw. möglicher Vereinfachungen ab. Diese beiden Parameter sind abhängig von der jeweiligen Repräsentation von Einschränkungs- und Zielsprache.

Anhand dieser Betrachtungen kommen Benedikt, Puppis und Riveros zu den in Tabelle 3.1 Komplexitätsergebnissen für die jeweiligen Repräsentationen von Einschränkungssprache R und Zielsprache T:

R \ T	Fix	DFA	NFA	LTL
Fix	Konstant	PTIME	PSPACE	PSPACE
DFA	PTIME	CoNP	PSPACE	PSPACE
NFA	PTIME	PTIME	PSPACE	PSPACE
LTL	PSPACE	PSPACE	PSPACE	CoNEXP

Tabelle 3.1: Komplexität des Bounded Repair Problems abhängig von der Repräsentation von Einschränkungs- und Zielsprache.

3.2 Bounded Streaming Repair

Für die Analyse des Bounded Streaming Repair Problems, zeigen Benedikt, Puppis und Riveros [BPR11] die Analogien zu Erreichbarkeitsspielen auf und übertragen Ergebnisse aus der Spieletheorie auf das Bounded Streaming Repair Problem. Ausgehend von einer Einschränkungs- und einer Zielsprache, jeweils repräsentiert durch einen deterministischen endlichen Automaten, konstruieren sie ein Erreichbarkeitsspiel zwischen *Adam* und *Eva*, welches genau dann von *Eva* gewonnen werden kann, wenn die Streaming Repair Distanz von der Einschränkungssprache zur Zielsprache endlich ist.

3.2.1 Erreichbarkeitsspiele

Für eine Einschränkungssprache R und eine Zielsprache T, repräsentiert durch zwei deterministische endliche Automaten $\mathcal{R} = (Q_R, \Sigma_R, \delta_R, q_{0,R}, F_R)$ bzw. $\mathcal{T} = (Q_T, \Sigma_T, q_{0,T}, F_T)$ definieren wir ein Erreichbarkeitsspiel $\Gamma_{\mathcal{R},\mathcal{T}} = (V, E, \rho)$, welches sich aus den starken Zusammenhangskomponenten der Automaten $\mathcal{R}$ und $\mathcal{T}$ wie folgt ergibt:

Die Knotenmenge V enthält alle Paare (C, D) und (D, C) aus starken Zusammenhangskomponenten $C \in SCC(\mathcal{R})$, $D \in SCC(\mathcal{T})$. Die Kantenmenge E der Spielearena $\Gamma_{\mathcal{R},\mathcal{T}}$ enthält für starke Zusammenhangskomponenten $C, C' \in SCC(\mathcal{R})$ und $D, D' \in SCC(\mathcal{T})$ die Kanten:

- $((D, C)(C', D))$ wenn es eine Kante (C, C') im Graph der starken Zusammenhangskomponenten $Dag(\mathcal{R})$ gibt.
- $((C, D)(D', C))$ wenn es eine Kante (D, D') im Graph der starken Zusammenhangskomponenten $Dag^*(\mathcal{T})$ gibt und $L(\mathcal{R}|C) \subseteq L(\mathcal{T}|D')$.

An Knoten der Form (C, D) ist *Adam* am Zug, Knoten der Form (D, C) sind *Eva*s Knoten.

Als Startknoten wählen wir die Position, die sich aus den starken Zusammenhangskomponenten der Startzustände der Automaten $\mathcal{R}$ und $\mathcal{T}$ ergibt: $v_0 = (\mathcal{C}(q_{0,T}), \mathcal{C}(q_{0,R}))$. An diesem Knoten ist *Eva* am Zug.

Die Idee des Erreichbarkeitsspiels $\Gamma_{\mathcal{R},\mathcal{T}}$ ist es, dass *Adam* einen Pfad in Graph $Dag(\mathcal{R})$ Knoten für Knoten vorgibt und *Eva* jeweils mit einem Knoten aus $Dag^*(\mathcal{T})$ antwortet, um somit einen Pfad in $Dag^*(\mathcal{T})$ zu finden, der den vorgegebenen Pfad überdeckt. Die erste starke Zusammenhangskomponente von $\mathcal{R}$ ist durch den Startzustand bestimmt, somit muss *Eva* als erste ziehen und zeigen, womit sie diese Zusammenhangskomponente überdecken kann. Benedikt, Puppis und Riveros haben in [BPR11] Theorem 3 folgende Analogie zwischen dem Streaming Repair Problem und dem so konstruierten Erreichbarkeitsspiels $\Gamma_{\mathcal{R},\mathcal{T}}$ aufgezeigt.

Satz 3.2 (Benedikt, Puppis, Riveros)**.** *Seien $\mathcal{R}$ und $\mathcal{T}$ zwei deterministische endliche Automaten, dann sind die folgenden Bedingungen äquivalent:*

- *Die Streaming Repair Distanz von $L(\mathcal{R})$ zu $L(\mathcal{T})$ ist endlich: $dist_{stream}(L(\mathcal{R}), L(\mathcal{T})) < \infty$.*
- *Eva hat eine Gewinnstrategie für das Erreichbarkeitsspiel $\Gamma_{\mathcal{R},\mathcal{T}}$ an der Startposition $v_0 = (\mathcal{C}(q_{0,T}), \mathcal{C}(q_{0,R}))$.*
- *Die Streaming Repair Distanz von $L(\mathcal{R})$ zu $L(\mathcal{T})$ ist begrenzt durch $(1 + |Dag(\mathcal{R})|) \cdot |\mathcal{T}|$.*

Eine wichtige Rolle spielt an dieser Stelle, dass sowohl $\mathcal{R}$ als auch $\mathcal{T}$ *deterministisch* sind, da somit an jeder Position eines Wortes w klar ist, in welchem Zustand sich ein zugehöriger Lauf von $\mathcal{R}$ bzw. $\mathcal{T}$ auf w befindet. Ähnlich wie in Abschnitt 3.2.1 beschrieben, existiert somit nach Konstruktion für einen Teilpfad $q_0 \xrightarrow{a_1} \ldots \xrightarrow{a_n} q_n$ innerhalb einer starken Zusammenhangskomponente C_R des Automaten $\mathcal{R}$ ein entsprechender Pfad $q'_0 \xrightarrow{a_1} \ldots \xrightarrow{a_n} q'_n$. Erfolgt innerhalb eines Laufes von $\mathcal{R}$ auf w ein Wechsel der starken Zusammenhangskomponente, so muss in $\mathcal{T}$ wieder eine Entsprechung gefunden werden. Im Gegensatz zum Bounded Repair Problem muss beim Bounded Streaming Repair Problem der entsprechende Pfad in $Dag^*(\mathcal{T})$ bereits konstruiert werden, während das Wort w gelesen wird. Hieraus ergibt sich die Spielsituation.

Beispiel 3.2. Abbildung 3.2 zeigt zwei Automaten $\mathcal{R}$ und $\mathcal{T}$ sowie das Erreichbarkeitsspiel $\Gamma_{\mathcal{R},\mathcal{T}}$, welches aus den Zusammenhangskomponenten $C_0 = \{q_0, q_1\}, C_1 = \{q_2\}, D_0 = \{q'_0\}$ und $D_1 = \{q'_1, q'_2\}$ mit Hilfe der obigen Konstruktion berechnet werden kann.

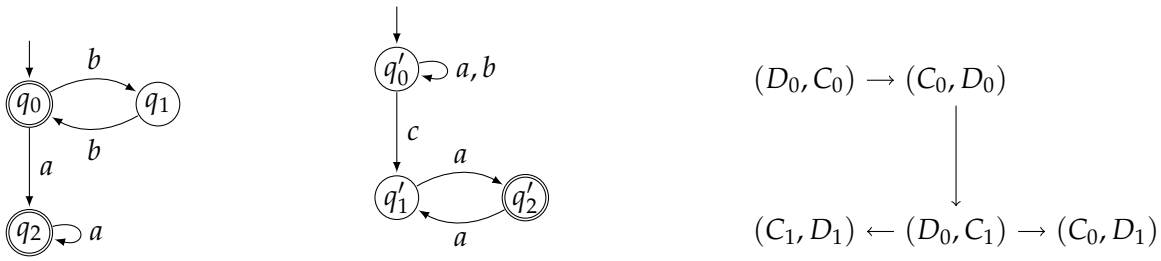


Abbildung 3.2: Zwei Automaten $\mathcal{R}$ und $\mathcal{T}$ sowie das daraus konstruierte Erreichbarkeitsspiel $\Gamma_{\mathcal{R},\mathcal{T}}$.

Offensichtlich besitzt *Eva* für das Spiel $\Gamma_{\mathcal{R},\mathcal{T}}$ eine Gewinnstrategie an der Startposition $v_0 = (D_0, C_0)$. Da $L(\mathcal{R}) \subseteq L(\mathcal{T})$ ist auch einfach zu sehen, dass $\text{dist}_{stream}(L(\mathcal{R}), L(\mathcal{T})) < \infty$.

3.2.2 Komplexität

Die Arena des Spiels $\Gamma_{\mathcal{R},\mathcal{T}}$ ist der in Abschnitt 3.2.1 angegebenen Konstruktion ein bipartiter, azyklischer, endlicher Graph. Somit können Gewinnstrategien für *Adam* bzw. *Eva* sowie die Positionen, an welchen *Adam* bzw. *Eva* eine Gewinnstrategie besitzen offensichtlich (durch entsprechendes Markieren der Knoten ausgehend von den finalen Knoten) in polynomieller Zeit ermittelt werden.

Die Komplexität des Bounded Streaming Repair Problems hängt somit im Wesentlichen von der Komplexität der Konstruktion des Spiels $\Gamma_{\mathcal{R},\mathcal{T}}$ ab und dessen Größe in Abhängigkeit von der Repräsentation der Sprachen R und T ab. Für deterministische endliche Automaten $\mathcal{R}$ und $\mathcal{T}$ kann das Spiel $\Gamma_{\mathcal{R},\mathcal{T}}$ in polynomieller Zeit berechnet werden. Ist einer der beiden Automaten nichtdeterministisch, so gestaltet sich das Problem deutlich schwieriger.

Anhand dieser und weiterer Betrachtungen kommen Benedikt, Puppis und Riveros bezüglich des Bounded Streaming Repair Problems zu den in Tabelle 3.2 dargestellten Komplexitätsergebnissen für die jeweiligen Repräsentationen von Einschränkungssprache R und Zielsprache T:

R \ T	Fix	DFA	NFA
Fix	Konstant	PTIME	PSPACE
DFA	PTIME	PTIME	PSPACE
NFA	PSPACE	PSPACE	PSPACE, EXPTIME

Tabelle 3.2: Komplexität des Bounded Streaming Repair Problems abhängig von der Repräsentation von Einschränkungs- und Zielsprache.

3.3 Zielsetzung

Sowohl das Bounded Repair Problem, als auch das Bounded Streaming Repair Problem kann man als Relation betrachten. Wir schreiben $L \leq_{\mathcal{BR}} L'$ (bzw. $L \leq_{\mathcal{BSR}} L'$), wenn die Distanz (bzw. die Streaming Repair Distanz) von der Sprache L zur Sprache L' endlich ist, *d.h.* $\text{dist}(L, L') < \infty$ (bzw. $\text{dist}(L, L') < \infty$). Gilt sowohl $L \leq_{\mathcal{BR}} L'$ als auch $L' \leq_{\mathcal{BR}} L$, so schreiben wir auch $L \; \mathcal{BR} \; L'$. Analog schreiben wir für $L \leq_{\mathcal{BSR}} L'$ und $L' \leq_{\mathcal{BSR}} L$ auch $L \; \mathcal{BSR} \; L'$.

Die Relationen $\leq_{\mathcal{BR}}$ und $\leq_{\mathcal{BSR}}$ sind reflexiv und transitiv, *d.h.* $\leq_{\mathcal{BR}}$ und $\leq_{\mathcal{BSR}}$ sind Quasiordnungen. $\mathcal{BR}$ und $\mathcal{BSR}$ sind somit reflexiv, transitiv und symmetrisch, *d.h.* es handelt sich dabei um Äquivalenzrelationen. Für eine Sprache L bezeichnen wir mit $[L]_{\mathcal{BR}}$ bzw. $[L]_{\mathcal{BSR}}$ deren Äquivalenzklasse.

Im Folgenden untersuchen wir, wie sich für einen Automaten $\mathcal{A}$ und die von $\mathcal{A}$ erkannte Sprache $L(\mathcal{A})$ die Äquivalenzklassen $[L(\mathcal{A})]_{\mathcal{BR}}$ und $[L(\mathcal{A})]_{\mathcal{BSR}}$ verhalten. Insbesondere untersuchen wir hierbei, inwieweit sich für einen Automaten $\mathcal{A}$ ein minimaler Automat $\mathcal{A}'$ mit $L(\mathcal{A}) \; \mathcal{BR} \; L(\mathcal{A}')$ bzw. $L(\mathcal{A}) \; \mathcal{BSR} \; L(\mathcal{A}')$ konstruieren lässt.

Bezüglich der Klassen Äquivalenzklassen von $\mathcal{BR}$ und $\mathcal{BSR}$ untersuchen wir zudem, welche Automatenmodelle zur Analyse und Beschreibung dieser Klassen geeignet sind. Dazu analysieren wir, welche Sprachklassen entsprechende Automatenmodelle beschreiben und welche alternativen Charakterisierung für diese Sprachklassen existieren.

4 Automatenmodelle und Sprachklassen

Sprachen und Sprachklassen können auf verschiedene Weisen charakterisiert werden. Die regulären Sprachen können beispielsweise dadurch charakterisiert werden, dass sie durch reguläre Ausdrücke, endliche Automaten oder durch ein reguläre Grammatik beschrieben werden können. Es gibt aber auch alternative, beispielsweise algebraische Charakterisierungen, für Sprachklassen.

In diesem betrachten Kapitel zunächst einige Sprachklassen, welche im Zusammenhang mit dem Bounded Repair Problem eine wichtige Rolle spielen. Insbesondere betrachten wir in Abschnitt 4.1 algebraische Charakterisierungen von Idealen und deren Booleschen Kombinationen innerhalb der regulären Sprachen. Wir geben zudem eine durch die Verbandstheorie motivierte Charakterisierung einiger Sprachklassen an. Für diese Sprachklassen betrachten wir in Abschnitt 4.2 einige Automatenmodelle, mit denen diese Sprachklassen vollständig beschrieben werden können. Die Ergebnisse dieser beiden Abschnitte wurden bereits in einer gemeinsamen Arbeit mit Manfred Kufleitner und Alexander Lauser veröffentlicht [JKL12].

Eine der wichtigen Fragestellungen für Sprachen, das Inklusionsproblem, betrachten wir dann in Abschnitt 4.3 für einen Teil der Automatenmodelle noch genauer. Hierbei gehen wir insbesondere auf den Zusammenhang zwischen dem Bounded Repair Problem, dem Bounded Streaming Repair Problem und dem Inklusionsproblem noch genauer ein.

4.1 Ideale und deren Boolesche Kombination

Für die Charakterisierung von Sprachen über unendlichen Worten spielt die Cantor-Topologie eine wichtige Rolle. Viele Eigenschaften ω-regulärer Sprachen können mit Hilfe der Cantor-Topologie beschrieben werden, siehe beispielsweise [Tho90, PP04]. Für Sprachen über endlichen Wörtern bilden rechtsseitige Ideale die Entsprechung zu offenen Mengen in der Cantor-Topologie und rechtsseitige Filter die Entsprechung zu geschlossenen Mengen in der Cantor-Topologie.

Für ein Monoid M ist $P \subseteq M$ ein rechtsseitiges (bzw. linksseitiges, beidseitiges) *Ideal*, wenn gilt $PM \subseteq P$ (bzw. $MP \subseteq P$, $MPM \subseteq P$). Das Komplement eines Ideals bezeichnen wir als *Filter*. Eine Sprache $L \subseteq \Sigma^*$ ist ein rechtsseitiges Ideal, wenn gilt $L \subseteq L\Sigma^*$. Eine Sprache $L \subseteq \Sigma^*$ ist ein rechtsseitiger Filter, wenn L unter Präfix abgeschlossen ist, *d.h.* wenn uv ein Wort aus L ist, dann ist auch u ein Wort aus L.

Die *Greenschen Relationen* sind auf einem Monoid wie folgt definiert. Für $x, y \in M$ ist $x \leq_{\mathcal{R}} y$ (bzw. $x \leq_{\mathcal{L}} y$, $x \leq_{\mathcal{J}} y$) wenn es $s, t \in M$ gibt mit $x = ys$ (bzw. $x = ty$, $x = tys$). Wir schreiben $x \mathcal{R} y$, wenn sowohl $x \leq_{\mathcal{R}} y$ als auch $y \leq_{\mathcal{R}} x$ gilt. Analog definieren wir die Relationen $\mathcal{L}$ und $\mathcal{J}$ bezüglich $\leq_{\mathcal{L}}$ und $\leq_{\mathcal{J}}$.

Eine Sprache $L \subseteq \Sigma^*$ wird durch einen Homomorphismus $h : \Sigma^* \to M$ *erkannt*, wenn es eine Menge $P \subseteq M$ gibt, für die L das Urbild ist, *d.h.* $L = h^{-1}(P)$ oder anders ausgedrückt $u \in L \Leftrightarrow h(u) \in P$.

Eine weiteres Hilfsmittel in der Beschreibung von Sprachklassen sind Verbandsidentitäten. Wir definieren hierzu induktiv ω-Terme über eine Menge von Variablen V. Jede Variable $x \in V$ ist ein ω-Term; und für zwei ω-Terme x und y sind auch xy und x^ω ω-Terme. Für eine Zahl $n \in \mathbb{N}$ definieren wir $x(n)$ induktiv durch $x(n) = x$ für $x \in V$ und $xy(n) = x(n)y(n)$, $(x^\omega)(n) = x(n)^{n!}$, *d.h.* $x(n)$ ist das Wort, das wir erhalten, wenn wir jeden in x vorkommenden Exponenten ω durch $n!$ ersetzen.

Mit diesen Hilfsmitteln können wir die Sprachklasse der Boolesche Kombinationen von rechtsseitigen Idealen wie folgt charakterisieren.

Satz 4.1. *Sei $L \subset \Sigma^*$ eine Sprache, die durch einen surjektiven Homomorphismus $h : \Sigma^* \to M$, welcher Σ^* auf einen endlichen Monoid M abbildet, erkannt wird. Dann sind die folgenden Aussagen äquivalent:*

1. *L ist eine Boolesche Kombination von rechtsseitigen (bzw. linksseitigen, beidseitigen) Idealen.*
2. *$h(L)$ ist eine Vereinigung von $\mathcal{R}$-Klassen (bzw. $\mathcal{L}$-Klassen, $\mathcal{J}$-Klassen).*
3. *L erfüllt die Verbandsidentität $z(xy)^\omega x \leftrightarrow z(xy)^\omega$ (bzw. die Identität $s(ts)^\omega z \leftrightarrow s(ts)^\omega$, die Identität $s(ts)^\omega z(xy)^\omega x \leftrightarrow (ts)^\omega z(xy)^\omega$).*

Beweis. Wir zeigen "1" $\Leftrightarrow$ "2" und "2" $\Leftrightarrow$ "3" für rechtsseitige Ideale. Der Beweis für linksseitige und beidseitige Ideale funktioniert analog.

"1" $\Rightarrow$ "2": Sei $L = \cup_{i=1}^{n} P_i \setminus Q_i$ für rechtsseitige Ideale $P_i\Sigma^* \subseteq P_i$ und $Q_i\Sigma^* \subseteq Q_i$. Wir betrachten $u, v \in \Sigma^*$ mit $h(u) \ \mathcal{R} \ h(v)$. Seien nun $x, y \in \Sigma^*$ mit $h(v) = h(ux)$ und $h(u) = h(vy)$. Sei ferner $u_j = u(xy)^j$ und $v_j = u_j x$. Nach Konstruktion gilt $h(u_j) = h(h)$ und $h(v_j) = h(v)$, zudem ist u_j ein Präfix von v_j und v_j ist wiederum ein Präfix von u_{j+1}. Nehmen wir nun an, dass $u \in L$ und somit $h(u) \in h(L)$, dann folgt daraus, dass auch alle $h(u_j) \in h(L)$ sind und somit gilt für alle $j \in \mathbb{N} : u_j \in L$. Folglich muss für jedes $j \in \mathbb{N}$ auch ein $i \in \{1, \ldots, n\}$ so existieren, dass $u_j \in P_i \setminus Q_i$. Nach dem Schubfachprinzip existieren $k, l \in \mathbb{N}, k < l$ mit $u_k, u_l \in P_i \setminus Q_i$ für ein $i \in \{1, \ldots, n\}$. Aus $P_i\Sigma^* \subseteq P_i$ folgt zudem $v_k = u_k x \in P_i$. Nehmen wir nun an, dass $v_k \in Q_i$, dann würde aus $Q_i\Sigma^* \subseteq Q_i$ folgen, dass $u_l \in Q_i$, was ein Widerspruch zu $u_l \in P_i \setminus Q_i$ wäre. Somit gilt $v_k \in P_i \setminus Q_i$ und somit $h(v) = h(v_k) \in h(L)$. Damit folgt aus $h(u) \ \mathcal{R} \ h(v)$ und $h(u) \in h(L)$, dass auch $h(v) \in h(L)$ ist und somit $h(L)$ eine Vereinigung von $\mathcal{R}$-Klassen ist.

"1" $\Leftarrow$ "2": Sei R eine $\mathcal{R}$-Klasse von M und betrachten wir die beiden rechtsseitige Ideal $R' = \{x \mid x \leq_{\mathcal{R}} R\}$ und $R'' = \{x \mid x <_{\mathcal{R}} R\}$. Dann gilt $h^{-1}(R) = h^{-1}(R') \setminus h^{-1}(R'')$. Da Ideale unter Homomorphismen und inversen Homomorphismen abgeschlossen sind, ist $h^{-1}(R)$ ein Boolesche Kombination von rechtsseitigen Idealen und da $h(L)$ eine Vereinigung von $\mathcal{R}$-Klassen ist, folgt somit, dass L eine Boolesche Kombination von rechtsseitigen Idealen ist.

"2" $\Rightarrow$ "3": Sei $h(L)$ eine Vereinigung von $\mathcal{R}$-Klassen. Für alle hinreichend großen $n \geq 1$ und für alle $x, y, z \in \Sigma$ gilt $h(z(xy)^n) \; \mathcal{R} \; h(z(xy)^n x)$. Somit gilt $z(xy)^n \in L \Leftrightarrow z(xy)^n x$, was zeigt, dass die Verbandsidentität $z(xy)^\omega x \leftrightarrow z(xy)^\omega$ von L erfüllt wird.

"2" $\Leftarrow$ "3": Sei $h(w) \; \mathcal{R} \; h(z) \in h(L)$. Dann existieren $x, y \in \Sigma^*$ mit $h(w) = h(zx)$ und $h(z) = h(wy) = h(zxy)$. Entsprechend gilt $h(z) = h(z(xy)^n)$ für alle $n \in \mathbb{N}$ und somit ist $z(xy)^n \in L$. Für hinreichend große n folgt aus der Verbandsidentität $z(xy)^\omega x \leftrightarrow z(xy)^\omega$, dass auch $z(xy)^n x \in L$ und somit $h(w) = h(z(xy)^n x) \in h(L)$. Somit gilt auch $w \in L$, was zeigt, dass $h(L)$ eine Vereinigung von $\mathcal{R}$-Klassen ist.

□

4.2 Automaten Modelle

Neben der Möglichkeit einer algebraischen Charakterisierung, können Sprachklassen auch über Automatenmodelle beschrieben werden, die zur Beschreibung der Sprachen aus dieser Klasse herangezogen werden können. Wir werden in diesem Abschnitt Automatenmodelle für die folgenden Sprachklassen angeben:

- rechtsseitige Ideale
- unter Präfix abgeschlossene Sprachen
- unter Infix abgeschlossene Sprachen
- Boolesche Kombinationen rechtsseitiger Ideale

4.2.1 Flipautomaten

Wir bezeichnen einen endlichen Automaten $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ als *Flipautomaten*, wenn es keine Übergänge von Endzuständen zu Nicht-Endzuständen gibt. Ein Flipautomat kann also innerhalb eines Laufes nur einmal von einem Nicht-Endzustand zu einem Endzustand "flippen". Formal bedeutet das, dass der Automat die Bedingung $\delta \cap F \times \Sigma \times (Q \setminus F) = \emptyset$ erfüllt oder anders ausgedrückt aus $q \in F$ und $q \xrightarrow{a} p$ folgt $p \in F$.

Für einen minimalen vollständigen Flipautomaten können wir annehmen, dass es nur einen Endzustand q_f gibt, welcher für alle Zeichen $a \in \Sigma$ einen Übergang $q_f \xrightarrow{a} q_f$ besitzt.

Proposition 4.1. *Sei $L \subseteq \Sigma^*$ ein Sprache, die von einem vollständigen deterministischen Automaten $\mathcal{A}$ erkannt wird. Dann sind die folgenden Aussagen äquivalent.*

1. *L ist ein rechtsseitiges Ideal.*
2. *$\mathcal{A}$ ist ein Flipautomat*
3. *L wird von einem vollständigen (nichtdeterministischen) Flipautomaten erkannt.*

Beweis. Die Äquivalenz von "1" und "2" wurde bereits von Paz und Pleg gezeigt [PP65].

Die Implikation "2" $\Rightarrow$ "3" ist trivial. Wir zeigen "3" $\Rightarrow$ "1":

Sei $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ ein vollständiger Flipautomat mit $L(\mathcal{A}) = L$. Sei $u \in L$ und $a \in \Sigma$ beliebig. Dann gibt es einen Zustand $q_0 \in Q_0$ und einen Zustand $p \in F$ mit $q_0 \xrightarrow{u} p$. Da $\mathcal{A}$ vollständig ist, gibt es mindestens einen Zustand $q \in Q$ für welchen ein Übergang $p \xrightarrow{a} q$ existiert. Da $\mathcal{A}$ zudem ein Flipautomat ist, folgt aus $p \in F$ und $p \xrightarrow{a} q$, dass auch $q \in F$. Daher ist $ua \in L(\mathcal{A})$ und somit $L\Sigma \subseteq L$ und folglich auch $L\Sigma^* \subseteq L$.

□

Bemerkung 4.1. Jede reguläre Sprache kann durch einen beliebigen (nichtdeterministischen nicht-vollständigen) Flipautomaten erkannt werden. Insbesondere kann jede reguläre Sprache durch einen nichtdeterministischen endlichen Automaten mit einem einzigen Endzustand ohne ausgehende Übergänge erkannt werden. Ein Automat mit diesen Eigenschaften ist ein Flipautomat.

4.2.2 Vollständig akzeptierende Automaten

Als *vollständig akzeptierende Automaten* bezeichnen wir endliche Automaten, bei welchen Zustandsmenge und Endzustandsmenge identisch sind, *d.h.* $F = Q$. Vollständig akzeptierende Automaten besitzen also keine nicht-akzeptierenden Zustände. Ein Wort kann folglich nur dadurch abgelehnt werden, dass es innerhalb eines Laufes auf dem Wort keinen weiteren Übergang mehr gibt.

Korollar 4.1. *Sei $L \subset \Sigma$ eine Sprache, die von einem deterministischen endlichen co-erreichbaren Automaten $\mathcal{A}$ erkannt wird, dann sind die folgenden Aussagen äquivalent.*

1. *L ist unter Präfix abgeschlossen*
2. *$\mathcal{A}$ ist ein vollständig akzeptierender Automat*
3. *L wird von einem (nichtdeterministischen) vollständig akzeptierenden Automaten erkannt.*

Beweis. "1" $\Rightarrow$ "2": Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ und sei $p \in Q \setminus F$. Da $\mathcal{A}$ co-erreichbar (und da wir Erreichbarkeit annehmen damit auch trim) ist, existiert ein Zustand $q \in F$ und $u, v \in \Sigma^*$ mit $q_0 \xrightarrow{u} p$ und $p \xrightarrow{v} q$. Somit ist $uv \in L$ und da L nach "1" unter Präfix abgeschlossen ist, ist auch $u \in L$ und da $\mathcal{A}$ deterministisch ist, folgt $p \in Q$.

Die Implikation "2" $\Rightarrow$ "3" ist trivial.

"3" $\Rightarrow$ "1": Sei $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ ein nichtdeterministischer endlicher Automat mit $L(\mathcal{A}) = L$ und sei $u \notin L$. Dann wird u nicht von $\mathcal{A}$ akzeptiert, *d.h.* es gibt keinen akzeptierenden Lauf von $\mathcal{A}$ auf u. Da $\mathcal{A}$ ein vollständig akzeptierender Automat ist, folgt daraus insbesondere, dass es keinen Lauf von $\mathcal{A}$ auf u gibt, *d.h.* für alle $q_0 \in Q_0$ und alle $q \in Q$ gilt, dass $(q_0, u, q) \notin \delta$. Damit folgt für alle $v \in \Sigma^*$, alle $q_0 \in Q_0$ und alle $q \in Q$, dass $(q_0, uv, q) \notin \delta$, *d.h.* $uv \notin L$. Das zeigt, dass $\Sigma^* \setminus L$ ein rechtsseitiges Ideal ist.

□

4.2.3 Pfadautomaten

Ein *Pfadautomat* ist ein Automat $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$, bei welchem jeder Zustand zugleich Start- und Endzustand ist, *d.h.* $Q = Q_0 = F$. Ein Pfadautomat erkennt ein Wort $u \in \Sigma^*$ genau dann, wenn es zwei Zustände $p, q \in Q$ gibt, welche durch einen Pfad $p \xrightarrow{u} q$ miteinander verbunden sind.

Korollar 4.2. *Sei $L \subseteq \Sigma*$ eine reguläre Sprache. Dann ist L genau dann unter Infix abgeschlossen, wenn L von einem Pfadautomaten erkannt wird.*

Für Automaten mit einer deterministischen Übergangsrelation geht Korollar 4.2 implizit aus der Arbeit von Avgustinovich und Fried [AF06] hervor. Wir zeigen hier, dass das Ergebnis auch im allgemeinen Fall gilt.

Beweis. "$\Rightarrow$": Da L unter Infix abgeschlossen ist, ist L insbesondere auch unter Präfix abgeschlossen. Somit existiert entsprechend Korollar 4.1 ein deterministischer endlicher vollständig akzeptierender Automat $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q)$ mit $L(\mathcal{A}) = L$. Wir zeigen, dass für den Pfadautomaten $\mathcal{A}' = (Q, \Sigma, \delta, Q, Q)$ gilt $L(\mathcal{A}') \subseteq L(\mathcal{A})$. Sei $v \in L(\mathcal{A}')$, dann existiert nach Konstruktion ein Pfad $p \xrightarrow{v} q$ in $\mathcal{A}$. Da wir Erreichbarkeit voraussetzen, existiert somit auch ein $u \in \Sigma^*$ mit $q_0 \xrightarrow{u} p \xrightarrow{v} q$ und somit gilt $uv \in L(\mathcal{A}) = L$. Nach Voraussetzung ist L unter Infix abgeschlossen und somit auch $v \in L$.

"$\Leftarrow$": Sei $\mathcal{A} = (Q, \Sigma, \delta, Q, Q)$ ein Pfadautomat mit $L(\mathcal{A}) = L$. Da sowohl $\mathcal{A}$ als auch der Automat, welcher durch invertieren der Kanten und Vertauschen von Start- und Endzuständen aus $\mathcal{A}$ konstruiert werden kann, Flipautomaten sind, sind sowohl L als auch die Umkehrung $L^r = \{a_1 \dots a_n \in \Sigma^* \mid a_i \in \Sigma, a_n \dots a_1 \in \Sigma\}$ von L nach Proposition 4.1 unter Präfix abgeschlossen. Daraus folgt, dass L unter Infix abgeschlossen ist.

□

4.2.4 Schwache Automaten und Staiger-Wagner Automaten

Staiger-Wagner Automaten über unendlichen Worten wurden ursprünglich dazu genutzt ω-reguläre Sprachen $L \subset \Sigma^\omega$ zu charakterisieren, für die sowohl L als auch $\Sigma^\omega \setminus L$ deterministisch sind [SW74]. Schwache Automaten wurden von Muller, Saoudi und Schupp [MSS92] für alternierende Baumautomaten vorgestellt.

Ein Automat $\mathcal{A}$ ist *schwach*, wenn für alle starken Zusammenhangskomponenten des Automaten gilt, dass entweder alle oder keiner der darin enthaltenen Zustände Endzustände sind. Ein Lauf auf einem schwachen Automaten kann also nur beschränkt oft von einem Endzustand in einen Nicht-Endzustand bzw. umgekehrt übergehen.

Ein *Staiger-Wagner Automat* besitzt eine andere Akzeptanzbedingung als gewöhnliche endliche Automaten. Ob ein Lauf wird von einem Staiger-Wagner Automaten akzeptiert wird oder nicht, hängt von der Menge der während des Laufs besuchten Zustände ab (unabhängig von der Anzahl der Besuche oder vom zuletzt besuchten Zustand). Ein Staiger-Wagner-Automat wird durch ein 5-Tupel $\mathcal{B} = (Q, \Sigma, \delta, q_0, \mathcal{F})$ mit $\mathcal{F} \subseteq 2^Q$ angegeben. Ein Lauf $\rho = q_0, q_1, \ldots, q_n$ wird von $\mathcal{B}$ genau dann akzeptiert, wenn $\{q_0, q_1, \ldots, q_n\} \in \mathcal{F}$. Ein Wort $w \in \Sigma^*$ wird von $\mathcal{B}$ akzeptiert, wenn es ein akzeptierenden Lauf von $\mathcal{B}$ auf w gibt.

Wir zeigen, dass deterministische Staiger-Wagner Automaten und deterministische schwache Automaten über endlichen Worten gleich mächtig sind und die Klasse der Booleschen Kombinationen von rechtsseitigen Idealen mit Hilfe dieser Automaten vollständig beschrieben werden kann. Hierzu zeigen wir zunächst, dass zu jedem schwachen Automaten ein entsprechender Staiger-Wagner Automat konstruiert werden kann, der dieselbe Sprache erkennt.

Lemma 4.1. *Sei $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ ein schwacher Automat. Dann existiert eine Staiger-Wagner Akzeptanzbedingung $\mathcal{F} \subseteq 2^Q$ für die der Staiger-Wagner Automat $\mathcal{B} = (Q, \Sigma, \delta, q_0, \mathcal{F})$ die Sprache $L(\mathcal{B}) = L(\mathcal{A})$ erkennt.*

Beweis. Sei $future(q)$ die Menge der von einem Zustand q aus erreichbaren Zustände, welche nicht sich nicht in einer Zusammenhangskomponente mit q befinden, *d.h.* $future(q) = \{p \in Q \mid \exists u \in \Sigma^* : q \xrightarrow{u} p \wedge \nexists v \in \Sigma^* : p \xrightarrow{v} q\}$. Wir können $\mathcal{F}$ nun wie folgt konstruieren:

$$\mathcal{F} = \{F_i \mid \exists q \in F \cap F_i : F_i \subseteq Q \setminus future(q)\}.$$

Nach Konstruktion muss zu jedem Lauf ρ, dessen Menge der besuchten Zustände einem F_i entspricht, in $\mathcal{A}$ eine starke Zusammenhangskomponente aus Endzuständen existieren, die von ρ besucht, aber nicht mehr verlassen wird. Ein Wort $w \in \Sigma^*$ wird somit von $\mathcal{B}$ genau dann akzeptiert, wenn es auch von $\mathcal{A}$ akzeptiert wird. Es gilt also $L(\mathcal{B}) = L(\mathcal{A})$.

□

Mit Hilfe dieser Konstruktion, können wir nun folgenden Satz zeigen:

Satz 4.2. *Sei $L \subseteq \Sigma^*$ eine Sprache, die durch einen deterministischen endlichen Automaten $\mathcal{A}$ akzeptiert wird. Dann sind die folgenden Aussagen äquivalent:*

1. *L ist eine Boolesche Kombination von rechtsseitigen Idealen*
2. *$\mathcal{A}$ ist schwach*
3. *L wird von einem deterministischen Staiger-Wagner Automaten erkannt*

Beweis. "1" $\Rightarrow$ "2": Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0,)$ ein deterministischer endlicher Automat und seien $p \in F, q \in Q$ mit $p \xrightarrow{u} q$ und $q \xrightarrow{v} p$ für $u, v \in \Sigma^*$. Dann gibt es ein $z \in \Sigma^*$ mit $q_0 \xrightarrow{z} p$. Für alle $n \in \mathbb{N}$ gilt somit $z(xy)^n \in L$. Da L eine Boolesche Kombination von rechtsseitigen Idealen ist, erfüllt L nach Satz 4.1 die Verbandsidentität $z(xy)^\omega \leftrightarrow z(xy)^\omega x$. Somit existiert ein $n \in \mathbb{N}$ mit $z(xy)^n x \in L$. Da $\mathcal{A}$ deterministisch ist, folgt aus $q_0 \xrightarrow{z(xy)^n x} q$ und $z(xy)^n x \in L$, dass $q \in F$ ein Endzustand ist. Es ist somit leicht zu sehen, dass $\mathcal{A}$ schwach ist.

"2" $\Rightarrow$ "3": Folgt direkt aus Lemma 4.1.

"3" $\Rightarrow$ "1": Sei $\mathcal{B} = (Q, \Sigma, \delta, q_0, \mathcal{F})$ ein deterministischer Staiger-Wagner Automat und seien $x, y, z \in \Sigma^*$. Sei zudem $n > |Q|$, $q_0 \xrightarrow{z} q_1$ und $q_i \xrightarrow{xy} q_{i+1}$ für $1 \leq i < n$. Nach dem Schubfachprinzip muss es $k, l \in \mathbb{N}$ mit $1 \leq k < l \leq n$ geben mit $q_k = q_l$. Insbesondere folgt daraus, dass für alle $m \in \mathbb{N}$ die Mengen der besuchten Zustände für einen Lauf von $\mathcal{B}$ auf $z(xy)^{l-1}(xy)^m$ und für einen Lauf von $\mathcal{B}$ auf $z(xy)^{l-1}(xy)^m x$ identisch sind. Somit gilt entsprechend der Akzeptanzbedingung für Staiger-Wagner Automaten, dass $z(xy)^n \in L \Leftrightarrow z(xy)^n x \in L$ .. Somit erfüllt L die Verbandsidentität $z(xy)^\omega \leftrightarrow z(xy)^\omega x$, was nach Satz 4.1 zeigt, dass L eine Boolesche Kombination von rechtsseitigen Idealen ist.

□

Bemerkung 4.2. Jede reguläre Sprache kann sowohl durch einen nichtdeterministischen schwachen Automaten, als auch durch einen nichtdeterministischen Staiger-Wagner Automaten erkannt werden. Für schwache Automaten ist dies bereits implizit aus Bemerkung 4.1 klar, da jeder Flipautomat auch ein schwacher Automat ist. Aus Lemma 4.1 folgt dasselbe unmittelbar auch für Staiger-Wagner Automaten.

4.3 Inklusionsproblem

Nachdem wir nun verschiedene Sprachklassen und Automatenmodelle eingeführt haben, betrachten wir nun das Inklusionsproblem für einige dieser Klassen etwas genauer. Das Inklusionsproblem spielt insbesondere für die Berechnung des Erreichbarkeitsspiels $\Gamma_{\mathcal{R},\mathcal{T}}$ und die Lösung des Bounded Repair Problems für zwei gegebene Automaten $\mathcal{R}$ und $\mathcal{T}$ eine wichtige Rolle.

Ein Teilproblem des Bounded Repair Problems ist die Fragestellung, ob für eine starke Zusammenhangskomponente C des Automaten $\mathcal{R}$ und für eine starke Zusammenhangskomponente C' des Automaten $\mathcal{T}$ gilt: $L(\mathcal{R}|C) \subseteq L(\mathcal{T}|C')$. Die dabei betrachteten Automaten $\mathcal{R}|C$ bzw. $\mathcal{T}|C'$ sind stark zusammenhängende Pfadautomaten. Wir verallgemeinern hier die Betrachtungen dieses Problems von Benedikt, Puppis und Riveros [BPR11] um damit allgemeine Aussagen über die entsprechenden Automatenmodelle, unabhängig von Bounded Repair Problem, zu treffen.

Lemma 4.2. *Für zwei stark zusammenhängende Pfadautomaten $\mathcal{A}$ und $\mathcal{B}$ sind die folgenden Aussagen äquivalent:*

1. $L(\mathcal{A}) \subseteq L(\mathcal{B})$
2. $L(\mathcal{A}) \leq_{\mathcal{BSR}} L(\mathcal{B})$
3. $L(\mathcal{A}) \leq_{\mathcal{BR}} L(\mathcal{B})$

Beweis. Die Implikationen "1" $\Rightarrow$ "2" und "2" $\Rightarrow$ "3" sind trivial. Wir zeigen "3" $\Rightarrow$ "1":

Nehmen wir an, dass $L(\mathcal{A}) \nsubseteq L(\mathcal{B})$. Dann existiert ein Wort $u \in L(\mathcal{A})$ mit $u \notin L(\mathcal{B})$. Sei nun $q_0 \xrightarrow{u} q_n$ ein akzeptierender Lauf von $\mathcal{A}$ auf u. Da $\mathcal{A}$ stark zusammenhängend ist, existiert ein v mit $q_n \xrightarrow{v} q_0$. Nach Konstruktion von u und v gilt $w_n = (uv)^n \in L(\mathcal{A})$ für alle $n \in \mathbb{N}$. Zudem gilt $\text{dist}(w_n, L(\mathcal{B})) \geq n$, da nach Voraussetzung $u \notin L(\mathcal{B})$ auf dem Teilwort u mindestens eine Edit-Operation notwendig ist. Somit existiert für jedes $k \in \mathbb{N}$ ein Wort w_{k+1} mit $\text{dist}(w_{k+1}, L(\mathcal{B})) > k$ und damit gilt $L(\mathcal{A}) \nleq_{\mathcal{BR}} L(\mathcal{B})$.

□

Proposition 4.2. *Für zwei stark zusammenhängende Pfadautomaten $\mathcal{A} = (Q_A, \Sigma_A, \delta_A, Q_A, Q_A)$ und $\mathcal{B} = (Q_B, \Sigma_B, \delta_B, Q_B, Q_B)$ mit deterministischen Übergangsfunktionen δ_A und δ_B liegt das Inklusionsproblem (gilt $L(\mathcal{A}) \subseteq L(\mathcal{B})$?) in PTIME.*

Beweis. Für die Pfadautomaten $\mathcal{A}$ und $\mathcal{B}$ können wir zwei deterministische Automaten $\mathcal{A}' = (Q_A, \Sigma_A, \delta_A, q_A, Q_A)$ und $\mathcal{B}' = (Q_B, \Sigma_B, \delta_B, q_B, Q_B)$ für ein $q_A \in Q_A$ und ein $q_B \in Q_B$ wählen. $L(\mathcal{A}') \leq_{\mathcal{BSR}} L(\mathcal{B}')$ gilt genau dann, wenn *Eva* eine Gewinnstrategie für das Spiel $\Gamma_{\mathcal{A}',\mathcal{B}'}$ besitzt. Nach Konstruktion von $\Gamma_{\mathcal{A}',\mathcal{B}'}$ gilt dies genau dann, wenn $L(\mathcal{A}'|\mathcal{C}(q_A)) \subseteq L(\mathcal{B}'|\mathcal{C}(q_B))$ gilt. Nach Konstruktion gilt zudem $\mathcal{A}'|\mathcal{C}(q_A) = \mathcal{A}$ und $\mathcal{B}'|\mathcal{C}(q_B) = \mathcal{B}$ und somit $L(\mathcal{A}) \subseteq L(\mathcal{B}) \Leftrightarrow L(\mathcal{A}') \leq_{\mathcal{BSR}} L(\mathcal{B}')$.

Benedikt, Puppis und Riveros haben in [BPR11] Korollar 13 gezeigt, dass das Bounded Streaing Repair Problem für zwei deterministische endliche Automaten PTIME vollständig ist und damit liegt auch das Inklusionsproblem für stark zusammenhängende Pfadautomaten mit deterministischer Übergangsfunktion in PTIME.

□

Bemerkung 4.3. Die wesentliche Aussage von Proposition 4.2 wurde von Benedikt, Puppis und Riveros für die starken Zusammenhangskomponenten der betrachteten Automaten direkt gezeigt, *d.h.* es wurde die Fragestellung $L(\mathcal{A}|C) \subseteq L(\mathcal{B}|D)$ für starke Zusammenhangskomponenten C und D betrachtet. Wir treffen die entsprechende Aussage für stark zusammenhängende Pfadautomaten und lösen somit das Ergebnis aus dem Zusammenhang des Bounded Streaming Repair Problems.

Proposition 4.3. *Für zwei (allgemeine) stark zusammenhängende Pfadautomaten $\mathcal{A}$ und $\mathcal{B}$ ist das Inklusionsproblem (gilt $L(\mathcal{A}) \subseteq L(\mathcal{B})$?) PSPACE vollständig.*

Beweis. Wir wissen, dass das Inklusionsproblem für nichtdeterministische endliche Automaten PSPACE vollständig ist (siehe [AH74]). Es existiert somit insbesondere ein PSPACE-Algorithmus um $L(\mathcal{A}) \subseteq L(\mathcal{B})$ für stark zusammenhängende Pfadautomaten zu entscheiden. Es genügt somit zu zeigen, dass das Inklusionsproblem für stark zusammenhängende Pfadautomaten PSPACE hart ist. Wir reduzieren hierzu das Inklusionsproblem für nichtdeterministische endliche Automaten auf das Inklusionsproblem für stark zusammenhängende Pfadautomaten.

Seien $\mathcal{A} = (Q_A, \Sigma, \delta_A, Q_{0A}, F_A)$ und $\mathcal{B} = (Q_B, \Sigma, \delta_B, Q_{0B})$ zwei beliebige nichtdeterministische endliche Automaten und sei $\alpha \notin \Sigma$ ein Zeichen, welches nicht im Alphabet Σ vorkommt. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass $\mathcal{A}$ und $\mathcal{B}$ sowohl erreichbar als auch co-erreichbar, also trim sind. Wir können somit zu $\mathcal{A}$ und $\mathcal{B}$ zwei Pfadautomaten $\mathcal{A}' = (Q_A, \Sigma', \delta'_A, Q_A, Q_A)$ und $\mathcal{B}' = (Q_B, \Sigma', \delta'_B, Q_B Q_B)$ mit $\Sigma' = \Sigma \cup \{\alpha\}$ und δ'_A, δ'_B wie folgt konstruieren:

δ'_A ergibt sich aus δ_A indem wir für jeden Startzustand $q_0 \in Q_{0A}$ und jeden Endzustand $q_f \in F_A$ einen Übergang $q_f \xrightarrow{\alpha} q_0$ hinzufügen. Für δ'_B gehen wir analog vor. Wir erhalten somit $\delta'_A = \delta_A \cup (F_A \times \{\alpha\} \times Q_{0A})$ und $\delta'_B = \delta_B \cup (F_B \times \{\alpha\} \times Q_{0B})$.

Ein Wort w wird nach Konstruktion von $\mathcal{A}'$ genau dann akzeptiert, wenn es ein Präfix $u \in \Sigma^*$ und ein Postfix $v \in \Sigma^*$ gibt mit $uwv = w_1 \alpha w_2 \alpha \dots \alpha w_n$ für $w_i \in L(\mathcal{A})$. Entsprechendes gilt für $\mathcal{B}'$. Es ist somit einfach zu sehen, dass $L(\mathcal{A}') \subseteq L(\mathcal{B}')$ genau dann gilt, wenn auch $L(\mathcal{A}) \subseteq L(\mathcal{B})$ gilt.

□

5 Minimierung

In Kapitel 3 haben wir das Bounded Repair Problem und das Bounded Streaming Repair Problem betrachtet. Darauf aufbauend haben wir die Relationen $\mathcal{BR}$ und BSR eingeführt. In Kapitel 4 haben wir verschiedene Sprach- und Automatenklassen analysiert. Aufbauend auf diesen Erkenntnissen, betrachten wir nun zu einem Automaten $\mathcal{A}$ die Äquivalenzklassen $[\mathcal{A}]_{\mathcal{BR}}$ und $[\mathcal{A}]_{\mathcal{BSR}}$. Insbesondere untersuchen wir, ob sich aus einem Automaten $\mathcal{A}$ innerhalb der entsprechenden Äquivalenzklasse effizient ein minimaler Automat $\mathcal{A}'$ mit $L(\mathcal{A}')$ $\mathcal{BR}$ $L(\mathcal{A})$ bzw. $L(\mathcal{A}')$ $\mathcal{BSR}$ $L(\mathcal{A})$ konstruieren lässt.

Für die Relation $\mathcal{BR}$ betrachten wir dieses Problem in Abschnitt 5.1, auf die Relation $\mathcal{BSR}$ gehen wir in Abschnitt 5.2 genauer ein.

5.1 Bounded Repair

Um das Bounded Repair Problem für zwei (nichtdeterministische) endliche Automaten $\mathcal{R}$ und $\mathcal{T}$ zu entscheiden, können wir entsprechend Satz 3.2 für alle Pfade π in $Dag(\mathcal{R})$ prüfen, ob ein Pfad π' in $Dag^*(\mathcal{T})$ existiert, der π überdeckt. Die Frage, ob die Distanz $\text{dist}(\mathcal{R}, \mathcal{T})$ endlich ist, hängt also von der Struktur der beiden Graphen und von den Sprachen ab, die für eine starke Zusammenhangskomponente $C \in SCC(\mathcal{R})$ von $\mathcal{R}|C$ bzw. für eine starke Zusammenhangskomponente $D \in SCC(\mathcal{T})$ von $\mathcal{T}|D$ erkannt wird.

Entsprechend dieser Beobachtung können wir das Minimierungsproblem bei beschränkter Repair Distanz für einen gegebenen Automaten $\mathcal{A}$ in zwei Teilprobleme zerlegen:

- Minimierung des gerichteten azyklischen Graphen der starken Zusammenhangskomponenten $Dag(\mathcal{A})$.
- Minimierung der starken Zusammenhangskomponenten.

Bevor wir das Minimierungsproblem genauer betrachten, analysieren wir zunächst die Eigenschaften der Äquivalenzklasse $[L(\mathcal{A})]_{\mathcal{BR}}$ für einen nichtdeterministischen endlichen Automaten $\mathcal{A}$.

5.1.1 Sprachklassen

Das Bounded Repair Problem kann entsprechend Satz 3.1 für zwei Automaten $\mathcal{R}$ und $\mathcal{T}$ mit Hilfe der Graphen $Dag(\mathcal{R})$ und $Dag^*(\mathcal{T})$ entschieden werden. Für beide Graphen ist es dabei unerheblich, welche der Zustände Endzustände sind, es ist lediglich notwendig, dass von jeder Zusammenhangskomponente aus ein Endzustand erreichbar ist. Da wir voraussetzen, dass alle Zustände erreichbar sind, spielt es für die Struktur zudem keine Rolle, bei welchen Zuständen es sich um Startzustände handelt.

Korollar 5.1. *Zu jedem endlichen Automaten $\mathcal{A}$ lässt sich ein ein Pfadautomat $\mathcal{A}'$ konstruieren mit $L(\mathcal{A})$ $\mathcal{BR}$ $L(\mathcal{A}')$. Insbesondere existiert für jede reguläre Sprache L eine unter Infix abgeschlossene reguläre Sprache mit L $\mathcal{BR}$ L'.*

Beweis. Für jeden Automaten $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ ist dessen Graph der starken Zusammenhangskomponenten $Dag(\mathcal{A})$ identisch mit dem Graph der starken Zusammenhangskomponenten $Dag(\mathcal{A}')$ des Automaten $\mathcal{A}' = (Q, \Sigma, \delta, Q, Q)$. Zudem gilt für jede starke Zusammenhangskomponente C des Automaten $\mathcal{A}$, dass $L(\mathcal{A}|C) = L(\mathcal{A}'|C)$. Aus der Charakterisierung des Bounded Repair Problems entsprechend Satz 3.1 folgt $L(\mathcal{A})$ $\mathcal{BR}$ $L(\mathcal{A}')$ und somit die Behauptung.

5.1.2 $Dag(\mathcal{A})$ Minimierung

Das Ziel der Minimierung des gerichteten azyklischen Graphen $Dag(\mathcal{A})$ innerhalb der Klasse $[L(\mathcal{A})]_{\mathcal{BR}}$ ist es, einen Automaten $\mathcal{A}'$ zu finden, für welchen die Anzahl der starken Zusammenhangskomponenten $|SCC(\mathcal{A}')|$ minimal ist. Tatsächlich lässt sich zeigen, dass bereits die Überprüfung, ob die Zahl der starken Zusammenhangskomponenten eines Automaten $\mathcal{A}$ minimal ist, ein PSPACE hartes Problem ist.

Satz 5.1. *Für einen nichtdeterministischen endlichen Automaten $\mathcal{A}$ ist es PSPACE hart zu entscheiden, ob ein Automaten $\mathcal{A}'$ mit $L(\mathcal{A}') \in [L(\mathcal{A})]_{\mathcal{BR}}$ und $|SCC(\mathcal{A}')| < |SCC(\mathcal{A})|$ existiert.*

Beweis. In Proposition 4.3 haben wir gezeigt, dass das Inklusionsproblem bereits für stark zusammenhängende Pfadautomaten PSPACE vollständig ist. Wir zeigen, dass für zwei stark zusammenhängende Pfadautomaten $\mathcal{B}$ und $\mathcal{B}'$ ein Automat $\mathcal{A}$ konstruiert werden kann, für welchen genau dann ein Automat $\mathcal{A}'$ mit $L(\mathcal{A}') \in [L(\mathcal{A})]_{\mathcal{BR}}$ und $|SCC(\mathcal{A}')| < |SCC(\mathcal{A})|$ existiert, wenn gilt $L(\mathcal{B}) \subseteq L(\mathcal{B}')$.

Seien $\mathcal{B} = (Q_B, \Sigma_B, \delta_B, Q_B, Q_B)$ und $\mathcal{B}' = (Q_{B'}, \Sigma_{B'}, \delta_{B'}, Q_{B'}, Q_{B'})$ zwei stark zusammenhängende Pfadautomaten und seien $\alpha \notin (\Sigma_B \cup \Sigma_B')$ sowie $p, q \notin (Q_B \dot\cup Q_{B'})$. Dann können wir daraus wie folgt einen Automaten $\mathcal{A} = (Q, \Sigma, \delta, Q, Q)$ konstruieren:

- Die Zustandsmenge Q ergibt sich aus den Zustandsmengen der beiden Automaten und den zwei weiteren Zuständen p und q, *d.h.* $Q = Q_B \dot\cup Q_{B'} \dot\cup \{p, q\}$.
- $\Sigma = \Sigma_B \cup \Sigma_{B'} \cup \{\alpha\}$.

- Die Übergangsrelation δ enthält alle Übergänge aus δ_B und δ'_B. Zusätzlich enthält δ die Übergänge $p \xrightarrow{\alpha} p$ und $q \xrightarrow{\alpha} q$. Für ein $q_b \in Q_B$ und ein $q_{B'} \in Q'_B$ fügen wir zudem die Übergänge $p \xrightarrow{\alpha} q_b$, $p \xrightarrow{\alpha} q_{b'}$ und $q_{b'} \xrightarrow{\alpha} q$ ein.

Nach Konstruktion besitzt der Automat $\mathcal{A}$ den in Abbildung 5.1 dargestellten Graph der starken Zusammenhangskomponenten.

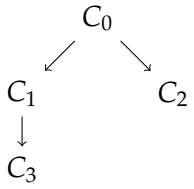


Abbildung 5.1: Graph der starken Zusammenhangskomponenten $Dag(\mathcal{A})$

Für die Zusammenhangskomponenten gilt $L(\mathcal{A}|C_0) = \alpha^* = L(\mathcal{A}|C_3)$ sowie $L(\mathcal{A}|C_1) = L(\mathcal{B}')$ und $L(\mathcal{A}|C_2) = L(\mathcal{B})$. Es ist einfach zu sehen, dass es zu $\mathcal{A}$ genau dann einen Automaten $\mathcal{A}'$ mit nur drei Zusammenhangskomponenten und $L(\mathcal{A}')\ \mathcal{BR}\ L(\mathcal{A})$ gibt, wenn $L(\mathcal{B}) \subseteq L(\mathcal{B}')$ gilt.

□

Satz 5.1 zeigt auch unmittelbar, die Komplexität der Suche eines Automaten $\mathcal{A}'$ mit minimalem gerichteten azyklischen Graphen $Dag(\mathcal{A}')$ innerhalb der Äquivalenzklasse $[L(\mathcal{A})]_{\mathcal{BR}}$ eines Automaten $\mathcal{A}$. Damit hat auch die Suche eines Automaten $\mathcal{A}'$ mit einer minimalen Anzahl von Zuständen mindestens dieselbe Komplexität.

Wir werden sehen, dass sich die entsprechenden Probleme für den Bounded Streaming Repair Fall effizienter lösen lassen.

5.2 Bounded Streaming Repair

Um für zwei deterministische endliche Automaten $\mathcal{R}$ und $\mathcal{T}$ zu entscheiden, ob die Streaming Repair Distanz $\text{dist}_{stream}(L(\mathcal{R}), L(\mathcal{T}))$ von $L(\mathcal{R})$ zu $L(\mathcal{T})$ endlich ist, können wir entsprechend Satz 3.2 das Erreichbarkeitsspiel $\Gamma_{\mathcal{R},\mathcal{T}}$ betrachten. Dieses Spiel ergibt sich aus den Graphen $Dag(\mathcal{R})$ und $Dag^*(\mathcal{T})$, wie in Abschnitt 3.2 beschrieben. Die Frage, ob die Streaming Repair Distanz von $\text{dist}_{stream}(L(\mathcal{R}), L(\mathcal{T}))$ endlich ist, hängt entsprechend dieser Charakterisierung sowohl von der Struktur der gerichteten azyklischen der starken Zusammenhangskomponenten $Dag(\mathcal{R})$ und $Dag(\mathcal{T})$, sowie den von den einzelnen Zusammenhangskomponenten erkannten Sprachen ab.

Entsprechend dieser Beobachtung können wir das Minimierungsproblem bei beschränkter Streaming Repair Distanz für einen gegebenen Automaten $\mathcal{A}$ in zwei Teilprobleme zerlegen:

- Minimierung des gerichteten azyklischen Graphen der starken Zusammenhangskomponenten $Dag(\mathcal{A})$.
- Minimierung der starken Zusammenhangskomponenten.

Bevor wir diese beiden Teilprobleme genauer analysieren, betrachten wir in Abschnitt 5.2.1 zunächst, wie sich für einen gegebenen Automaten $\mathcal{A}$ die Äquivalenzklasse $[L(\mathcal{A})]_{\mathcal{BSR}}$ im Bezug auf die in Kapitel 4 betrachteten Sprachklassen verhält.

Da die starken Zusammenhangskomponenten deterministische Übergangsfunktionen besitzen, können auf diese gängige Algorithmen zur Minimierung deterministischer endlicher Automaten angewandt werden. Wie wir in Korollar 5.2 zeigen, können wir hierbei annehmen, dass alle Zustände der starken Zusammenhangskomponente Endzustände sind. Es bleibt also das Problem der Minimierung des gerichteten azyklischen Graphen der starken Zusammenhangskomponenten. Dieses Problem betrachten wir in Abschnitt 5.2.2 genauer.

5.2.1 Sprachklassen

Für zwei deterministische endliche Automaten $\mathcal{R}$ und $\mathcal{T}$ kann zur Entscheidung des Bounded Streaming Repair Problems entsprechend Satz 3.2 das Spiel $\Gamma_{\mathcal{R},\mathcal{T}}$ betrachtet werden. Diese Konstruktion führt uns unmittelbar zu folgendem Korollar.

Korollar 5.2. *Zu jedem endlichen Automaten $\mathcal{A}$ existiert ein vollständig akzeptierender Automat $\mathcal{A}'$ mit $L(\mathcal{A})$ $\mathcal{BSR}$ $L(\mathcal{A}')$. Insbesondere existiert für jede reguläre Sprache L eine unter Präfix abgeschlossene reguläre Sprache L' mit L $\mathcal{BSR}$ L'.*

Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ ein endlicher Automat. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass $\mathcal{A}$ deterministisch und trim ist. Sei nun $\mathcal{A}' = (Q, \Sigma, \delta, Q_0, Q)$, so gilt insbesondere $L(\mathcal{A}) \subseteq L(\mathcal{A}')$ und somit $L(\mathcal{A}) \leq_{\mathcal{BSR}} L(\mathcal{A}')$. Es bleibt also zu zeigen, dass $L(\mathcal{A}) \leq_{\mathcal{BSR}} L(\mathcal{A}')$.

Da es für die Konstruktion des Erreichbarkeitsspiels $\Gamma_{\mathcal{A}',\mathcal{A}}$ keine Rolle spielt, welche Zustände der Automaten $\mathcal{A}$ und $\mathcal{A}'$ Endzustände sind, sondern lediglich betrachtet wird, von welchen Zusammenhangskomponenten aus ein Endzustand erreichbar ist, sind die Spielegraphen $\Gamma_{\mathcal{A}',\mathcal{A}}$ und $\Gamma_{\mathcal{A},\mathcal{A}'}$ isomorph. *Eva* besitzt für $\Gamma_{\mathcal{A}',\mathcal{A}}$ folglich eine Gewinnstrategie, wenn sie eine Gewinnstrategie für $\Gamma_{\mathcal{A},\mathcal{A}'}$ besitzt, *d.h.* es gilt $L(\mathcal{A}) \leq_{\mathcal{BSR}} L(\mathcal{A}') \Leftrightarrow L(\mathcal{A}') \leq_{\mathcal{BSR}} L(\mathcal{A})$. Somit folgt $L(\mathcal{A})$ $\mathcal{BSR}$ $L(\mathcal{A}')$.

□

5.2.2 $Dag(\mathcal{A})$ Minimierung

In Abschnitt 3.2 wurde gezeigt, wie zu zwei deterministischen endlichen Automaten $\mathcal{R}$ und $\mathcal{T}$ ein Spiel $\Gamma_{\mathcal{R},\mathcal{T}}$ konstruiert werden kann, mit Hilfe dessen wir bestimmen können, ob die Streaming Repair Distanz von $\mathcal{R}$ zu $\mathcal{T}$ endlich ist.

Analog können wir für einen einzelnen deterministischen endlichen Automaten $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ ein Erreichbarkeitsspiel $\Gamma_{\mathcal{A}} = (V, E, \rho)$ wie folgt konstruieren:

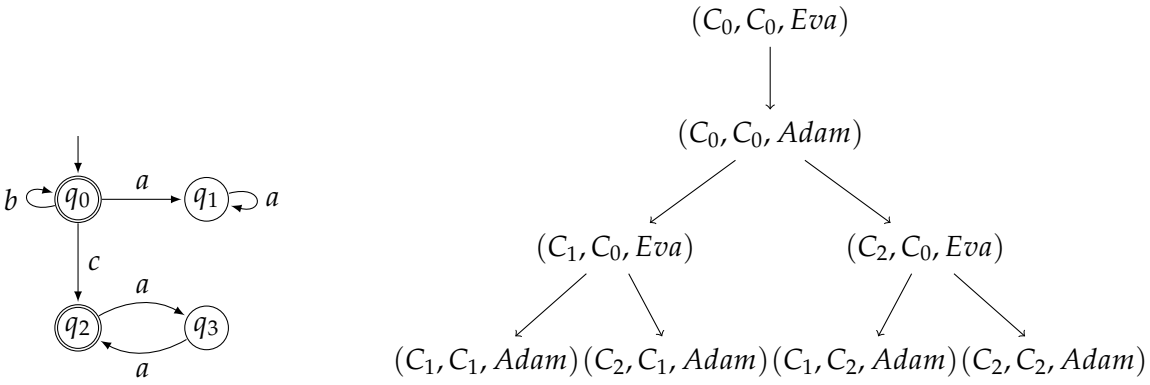
Als Knotenmenge wählen wir $V = SCC(Q) \times SCC(Q) \times \{Adam, Eva\}$, die Kantenmenge enthält für $C, D, C', D' \in SCC(Q)$ die Kanten

- $((D, C, Adam), (C', D, Eva))$ wenn in $Dag(\mathcal{A})$ eine Kante (C, C') existiert.
- $((C, D, Eva), (D', C, Adam))$ wenn in $Dag^*(\mathcal{A})$ eine Kante (D, D') existiert und wenn zudem gilt $L(\mathcal{A}|C) \subseteq L(\mathcal{A}|D)$.

An Knoten der Form $(C, D, Adam)$ ist *Adam* am Zug, Knoten der Form (D, C, Eva) sind *Eva*s Knoten. Die dritte Komponente gibt also den Spieler an, die erste Komponente enthält immer die Position des Gegenspielers im Graphen $Dag(\mathcal{A})$ *bzw.* $Dag^*(\mathcal{A})$, und die zweite Komponente gibt die eigene Position im jeweils anderen Graphen an.

Nach Konstruktion besitzt *Eva* für das Erreichbarkeitsspiel $\Gamma_{\mathcal{A}}$ eine Gewinnstrategie an der Startposition $v_0 = (\mathcal{C}(q_0), \mathcal{C}(q_0), Eva)$. Da für jede Kante (C_i, C_j) im Graphen $Dag(\mathcal{A})$ eine entsprechende Kante (C_i, C_j) im Graphen $Dag^*(\mathcal{A})$ existiert, besitzt *Eva* insbesondere eine triviale Gewinnstrategie, bei welcher *Eva* für jeden Wahl einer starken Zusammenhangskomponente C_i in $Dag(\mathcal{A})$ dieselbe Zusammenhangskomponente in $Dag^*(\mathcal{A})$ wählt, *d.h. Eva* zieht von der Startposition $(\mathcal{C}(q_0), \mathcal{C}(q_0), Eva)$ zu $(\mathcal{C}(q_0), \mathcal{C}(q_0), Adam)$ und jeden Zug $(C_i, C_i, Adam) \to (C_j, C_i, Eva)$ beantwortet *Eva* mit einem Zug $(C_j, C_i, Eva) \to (C_j, C_j, Adam)$.

Beispiel 5.1. Abbildung 5.2 zeigt einen Automaten $\mathcal{A}$ mit den starken Zusammenhangskomponenten $C_0 = \{q_0\}$, $C_1 = \{q_1\}$ und $C_2 = \{q_2, q_3\}$ mit dem daraus konstruierten Erreichbarkeitsspiel $\Gamma_{\mathcal{A}}$.



(a) $\mathcal{A}$ (b) $\Gamma_{\mathcal{A}}$

Abbildung 5.2: Ein Automat $\mathcal{A}$ und das daraus konstruierte Erreichbarkeitsspiel $\Gamma_{\mathcal{A}}$.

Es ist einfach zu sehen, dass zu Automat $\mathcal{A}$ in Abbildung 5.2 ein Automat $\mathcal{A}'$ mit nur zwei starken Zusammenhangskomponenten und $L(\mathcal{A}')$ $\mathcal{BSR}$ $L(\mathcal{A})$ konstruiert werden kann.

Unser Ziel ist es, für einen gegebenen Automaten $\mathcal{A}$ mit Hilfe des Spiels $\Gamma_{\mathcal{A}}$ einen Automaten $\mathcal{A}'$ mit einem minimalen Graphen $Dag(\mathcal{A}')$ zu konstruieren, für welchen gilt, $L(\mathcal{A}')\ \mathcal{BSR}\ L(\mathcal{A})$. Hierzu betrachten wir zunächst zwei Eigenschaften von $\mathcal{A}$, die man an mit Hilfe des Spielegraphen $\Gamma_{\mathcal{A}}$ feststellen kann:

1. *Äquivalente Zusammenhangskomponenten*:
   $\mathcal{A}$ besitzt zwei bezüglich der Relation $\mathcal{BSR}$ äquivalente Zusammenhangskomponenten.
2. *Beliebigkeit*:
   Eva besitzt für $\Gamma_{\mathcal{A}}$ an der Startposition v_0 eine nicht-triviale Gewinnstrategie.

Zwei Zusammenhangskomponenten C_i und C_j des Automaten $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ betrachten wir hierbei als äquivalent, wenn für ein $p \in C_i$ und ein $q \in C_j$ für die von den Automaten $\mathcal{A}' = (Q, \Sigma, \delta, p, F)$ und $\mathcal{A}'' = (Q, \Sigma, \delta, q, F)$ gilt $L(\mathcal{A}')\ \mathcal{BSR}\ L(\mathcal{A}'')$. C_i und C_j sind also äquivalent, wenn die bei C_i bzw. C_j beginnenden Teilautomaten von $\mathcal{A}$ in einer Äquivalenzklasse von $\mathcal{BSR}$ sind.

Unsere Idee ist es nun, zu einem gegebenen Automaten $\mathcal{A}$ einen Automaten $\mathcal{A}'$ zu konstruieren, für welchen *Eva* genau eine Gewinnstrategie für $\Gamma_{\mathcal{A}'}$ besitzt und welcher keine äquivalenten Zusammenhangskomponenten besitzt. Wir werden zeigen, dass für einen solchen Automaten der gerichtete azyklische Graph der Zusammenhangskomponenten $Dag(\mathcal{A}')$ minimal ist.

Äquivalente Zusammenhangskomponenten

Für einen deterministischen Automaten $\mathcal{A}$ sind zwei starke Zusammenhangskomponenten C_i und C_j äquivalent, wenn die bei C_i bzw. C_j beginnenden Teilautomaten von $\mathcal{A}$ in einer Äquivalenzklasse von $\mathcal{BSR}$ sind. Dies ist genau dann der Fall, wenn Eva für das Spiel $\Gamma_{\mathcal{A}}$ sowohl an der Position (C_i, C_j, Eva) als auch an der Position (C_j, C_i, Eva) eine Gewinnstrategie besitzt. Somit kann man äquivalente Zusammenhangskomponenten dem Spiel $\Gamma_{\mathcal{A}}$ direkt entnehmen.

Für einen deterministischen endlichen Automaten $\mathcal{A}$ kann sowohl das Erreichbarkeitsspiel $\Gamma_{\mathcal{A}}$, als auch die zugehörigen Gewinnstrategien für *Adam* und *Eva* in polynomieller Zeit berechnet werden (siehe Abschnitt 3.2.2). Entsprechend können wir zu einem gegebenen Automaten $\mathcal{A}$ entsprechend Algorithmus 5.1 in polynomieller Zeit einen Automaten $\mathcal{A}'$ konstruieren, welcher keine äquivalenten starken Zusammenhangskomponenten besitzt.

Für zwei äquivalente Zusammenhangskomponenten C und D wählt Algorithmus 5.1 die Zusammenhangskomponente C (zufällig) aus und entfernt alle Übergänge in $\mathcal{A}$, die zu einem der Zustände in der Zusammenhangskomponente C führen. Diese Übergänge werden durch Übergänge zu einem der Zustände aus der Zusammenhangskomponente D ersetzt.

Werden zwei äquivalente Zusammenhangskomponenten C und D zusammengelegt, so existieren für $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ zwei deterministische endliche Automaten $\mathcal{A}_C = (Q, \Sigma, \delta, p, F)$ und $\mathcal{A}_D = (Q, \Sigma, \delta, q, F)$ mit $p \in C$, $q \in D$ und $L(\mathcal{A}_C)\ \mathcal{BSR}\ L(\mathcal{A}_D)$, *d.h. Eva* besitzt sowohl für das Spiel $\Gamma_{\mathcal{A}_C,\mathcal{A}_D}$ als auch für das Spiel $\Gamma_{\mathcal{A}_D,\mathcal{A}_C}$ eine Gewinnstrategie. Damit kann auch

Algorithmus 5.1 Entfernen äquivalenter Zusammenhangskomponenten

```
function removeEquivalentSCCs(A)
    let A = (Q, Σ, δ, q_0, F)
    δ' ← δ
    equivalent ← equivalentSCCs(A)
    for (C, D) ∈ equivalent do
        let p ∈ D
        for (q, a, q') ∈ δ' and q' ∈ C do
            δ' ← δ' \ {(q, a, q')}
            δ' ← δ' ∪ {(q, a, p)}
        end for
    end for
    A' ← (Q, Σ, δ', q_0, F)
    return removeUnreachableStates(A')
end function
```

für die Spiele $\Gamma_{\mathcal{A},\mathcal{A}'}$ und $\Gamma_{\mathcal{A}',\mathcal{A}}$ eine Gewinnstrategie für *Eva* wie folgt konstruiert werden. Vom Startknoten aus wählt *Eva* für jede von Adam gewählte Zusammenhangskomponente C im Graphen $Dag(\mathcal{A})$ (bzw. $Dag(\mathcal{A}')$ dieselbe Zusammenhangskomponente, sofern diese in $Dag^*(\mathcal{A})$ (bzw. $Dag^*(\mathcal{A}')$) existiert. Andernfalls existiert in $Dag^*(\mathcal{A})$ (bzw. in $Dag^*(\mathcal{A}')$) eine zu C äquivalente starke Zusammenhangskomponente D, welche zu C äquivalent ist. Von da an kann *Eva* der Gewinnstrategie für das Spiel $\Gamma_{\mathcal{A}_C,\mathcal{A}_D}$ folgen.

Beliebigkeit

Für jedes zu einem deterministischen Automaten $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ gehörende Erreichbarkeitsspiel $\Gamma_{\mathcal{A}}$ besitzt *Eva* eine triviale Gewinnstrategie, *d.h. Eva* zieht von der Startposition $(\mathcal{C}(q_0), \mathcal{C}(q_0), Eva)$ zu $(\mathcal{C}(q_0), \mathcal{C}(q_0), Adam)$ und jeden Zug $(C_i, C_i, Adam) \to (C_j, C_i, Eva)$ beantwortet sie mit einem Zug $(C_j, C_i, Eva) \to (C_j, C_j, Adam)$.

Eine Gewinnstrategie τ_E für *Eva* ist nicht trivial, wenn sie von diesem Verhalten abweicht, *d.h.* wenn ein mit τ_E konformes Spiel $s = v_0 \dots v_n$ existiert, in welchem ein Knoten $v_k = (C_i, C_j, Adam)$ mit $C_i \neq C_j$ vorkommt. In diesem Fall kann *Eva* zwischen (mindestens) zwei Strategien wählen. Unser Ziel ist es, einen Automaten $\mathcal{A}'$ zu konstruieren, zu welchem *Eva* für $\Gamma_{\mathcal{A}'}$ keine nicht-triviale Gewinnstrategie besitzt und für welchen gilt $L(\mathcal{A}')\ \mathcal{BSR}\ L(\mathcal{A})$.

Nehmen wir nun an, dass *Eva* für das Erreichbarkeitsspiel $\Gamma_{\mathcal{A}}$ eine nicht-triviale Gewinnstrategie τ_E besitzt. Dann existiert ein mit der Gewinnstrategie τ_E konformes Spiel $s = v_0 \dots v_n$ mit $v_k = (C_i, C_j, Adam), C_i \neq C_j$ für ein $k \in \{1 \dots n\}$. Wählen wir nun $k \in \{1 \dots n\}$ minimal, so dass $v_k = (C_i, C_j, Adam), C_i \neq C_j$ gilt, so existieren für der Knoten v_{k-1} zwei Nachfolger, für welche *Eva* eine Gewinnstrategie besitzt, nämlich $v_k = (C_i, C_j, Adam)$ und $v'_k = (C_j, C_j, Adam)$. Wir gehen nun so vor, dass wir die entsprechenden Verzweigungen in $\Gamma_{\mathcal{A}}$ und einen Automaten $\mathcal{A}'$ konstruieren, für welchen *Eva* an der entsprechenden Stelle im Graph $\Gamma'_{\mathcal{A}}$ nicht aus zwei Möglichkeiten wählen kann.

Für die Konstruktion eines entsprechenden Automaten $\mathcal{A}'$, müssen die Mehrdeutigkeiten gefunden, analysiert und anschließend eliminiert werden.

Algorithmus 5.2 Finden von Beliebigkeiten

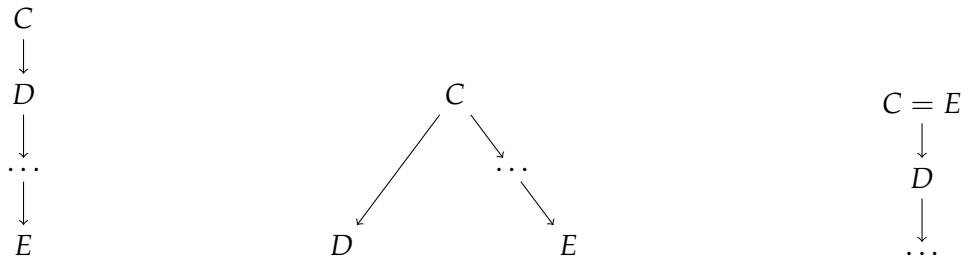
```
function findAmbiguity(Γ_A, v)
    let v = (C, D, player)
    if player = Adam then
        for v' ∈ successors(v, Γ_A) do
            ambiguity ← findAmbiguity(Γ_A, v')
            if amiguity ≠ NULL then
                return ambiguity
            end if
        end for
    else
        for v' ∈ successors(v, Γ_A) do
            let v' = (C', D', player')
            if v' ∈ EvesWinningNodes(Γ_A)  and C' ≠ D' then
                return (v, v')
            else if C' = D' then
                ambiguity ← findAmbiguity(Γ_A, v')
                if amiguity ≠ NULL then
                    return ambiguity
                end if
            end if
        end for
    end if
    return NULL
end function
```

Um diese Verzweigungen zu finden, führen wir in Algorithmus 5.2, beginnend von einem Knoten v, eine Tiefensuche auf $\Gamma_{\mathcal{A}}$ durch, welche berücksichtigt, dass *Eva* nur Pfade wählt, auf welchen sie gewinnen kann. Die Funktion findAmbiguity gibt dabei die erste gefundene Verzweigung mit dem Vorgänger im Spielegraph zurück. Da der Spielegraph azyklisch und endlich ist, terminiert der Algorithmus immer in polynomieller Zeit in Abhängigkeit von der Größe der Arena $\Gamma_{\mathcal{A}}$.

Die berechneten Beliebigkeiten bezüglich der Gewinnstrategie von *Eva* im Spielegraph $\Gamma_{\mathcal{A}}$ können wir nun Anhand der zu Grunde liegenden Struktur im Graph der der starken Zusammenhangskomponenten $Dag(\mathcal{A})$ unterscheiden. Die Funktion findAmbiguity in Algorithmus 5.2 liefert uns dazu zwei Knoten (D, C, Eva) und $(E, D, Adam)$. Nach Konstruktion des Erreichbarkeitsspiels $\Gamma_{\mathcal{A}}$ wissen wir, dass der Knoten E im Graph der starken Zusammenhangskomponenten $Dag(\mathcal{A})$ entweder identisch mit C ist, oder aber ein (nicht unbedingt direkter) Nachfolger von C ist. Entsprechend der Konstruktion des Algorithmus, wissen wir zudem, dass der Vorgängerknoten von (D, C, Eva) im Spielegraph $\Gamma_{\mathcal{A}}$ entweder der Knoten $(C, C, Adam)$ ist, oder $C = D = \mathcal{C}(q_0)$ gilt. Enthält C nicht den Startzustand

($q_0 \notin C$), so können wir daraus wiederum schließen, dass C der ein direkter Vorgänger von D im Graph $Dag(\mathcal{A})$ ist.

Für den Fall, dass $q_0 \notin C$ können wir somit drei verschieden Konstellationen bezüglich der Anordnung von C, D und E innerhalb des Graphen $Dag(\mathcal{A})$ unterscheiden. Diese sind in Abbildung 5.3 dargestellt.



(a) E Nachfolger von D (b) E parallel zu D (c) C und E sind identisch

Abbildung 5.3: Strukturen des Graphen $Dag(\mathcal{A})$ für Beliebigkeiten

Die einfachste Möglichkeit besteht darin, dass E ein Nachfolger von D und somit von D aus erreichbar ist (Abbildung 5.3(a)). Alternativ besteht die Möglichkeit, dass E von D aus nicht erreichbar ist, aber ein Pfad von C nach E in $Dag(\mathcal{A})$ existiert (Abbildung 5.3(b)). Die dritte Möglichkeit besteht darin, dass C und E identisch sind und D somit ein direkter Nachfolger von E ist (Abbildung 5.3(c)).

Für alle drei Konstellationen können wir, abhängig davon um welche es sich handelt, die Wahlmöglichkeit in der Gewinnstrategie für *Eva* auf einfache Weise eliminieren:

(a) E ist ein Nachfolger von D: In diesem Fall kann die starke Zusammenhangskomponente D übersprungen werden. Wir entfernen alle Übergänge $p \xrightarrow{a} q$ mit $p \in C$ und $q \in D$ aus der Übergangsrelation δ. Sollte anschließend kein Pfad $C \to \ldots \to E$ mehr in $Dag(\mathcal{A})$ existieren, so fügen wir für $p \in C$, $q \in E$ einen Übergang $p \xrightarrow{a} q$ ein.

(b) D und E sind von C aus erreichbar: In diesem Fall kann die Verbindung zwischen C und D übersprungen werden, da es einen alternativen Pfad über E gibt. Wir entfernen alle Übergänge $p \xrightarrow{a} q$ mit $p \in C$ und $q \in D$ aus der Übergangsrelation δ.

(c) C und E sind identisch: Die Zusammenhangskomponente D kann übersprungen werden. Wir entfernen daher alle Übergänge $p \xrightarrow{a} q$ mit $p \in C$ und $q \in D$ aus der Übergangsrelation δ. Anschließend fügen wir für alle direkten Nachfolger F von D (falls notwendig) einen Übergang von C nach F ein.

Für den Fall $q_0 \in C$ können wir statt q_0 einen neuen Startzustand $q \in E$ wählen.

Auf diese Weise können wir einzelne Wahlmöglichkeiten eliminieren. Anschließend können Zustände, die dadurch nicht mehr erreichbar sind, entfernt werden.

Bei jeder der Änderungsoperationen folgt nach Konstruktion für den resultierenden Automaten $\mathcal{A}'$, dass *Eva* sowohl für das Spiel $\Gamma_{\mathcal{A},\mathcal{A}'}$, als auch für das Spiel $\Gamma_{\mathcal{A}'}$ eine Gewinnstrategie besitzt, *d.h.* $L(\mathcal{A}')\ \mathcal{BSR}\ L(\mathcal{A})$. Die Gewinnstrategie für $\Gamma_{\mathcal{A},\mathcal{A}'}$ entspricht genau der nichttrivialen Gewinnstrategie für $\Gamma_{\mathcal{A}}$, die der Wahlmöglichkeit zu Grunde lag. Die Gewinnstrategie für $\Gamma\mathcal{A}', \mathcal{A}$ ergibt sich daraus, dass jeder Pfad in $Dag(\mathcal{A}')$ auch in $Dag^*(\mathcal{A})$ existiert und somit *Eva* auf jeden Zug von *Adam* in $Dag(\mathcal{A}')$ dieselbe Zusammenhangskomponente in $Dag^*(\mathcal{A})$ wählen kann.

Minimierungsalgorithmus

Setzt man nun die einzelnen Schritte zusammen, so erhält man folgenden Algorithmus zur Berechnung eines Automaten $\mathcal{A}'$ ohne Beliebigkeiten und äquivalente Zusammenhangskomponenten auf Eingabe eines Automaten $\mathcal{A}$. Wir werden zeigen, dass der daraus resultierende Automat mit der minimalen Anzahl von starken Zusammenhangskomponenten auskommt.

Algorithmus 5.3 Algorithmus zur Minimierung des gerichteten azyklischen Graphen $Dag(\mathcal{A})$

```
function minimizeDAG(A)
    A' ← removeEquivalentSCCs(A)
    let A' = (Q, Σ, δ, q_0, F)
    Γ_{A'} ← reachabilityGame(A')
    ambiguity ← findAmbiguity(Γ_{A'}, (C(q_0), C(q_0), Eva))
    while ambiguity ≠ NULL do
        A' ← removeAmbiguity(A', ambiguity)
        Γ_{A'} ← reachabilityGame(A')
        ambiguity ← findAmbiguity(Γ_{A'}, (C(q_0), C(q_0), Eva))
    end while
    return A'
end function
```

Mit Hilfe dieses Algorithmus können wir folgenden Satz zeigen:

Satz 5.2. *Zu jedem deterministischen endlichen Automat $\mathcal{A}$ kann in polynmieller Zeit ein deterministischer endlicher Automat $\mathcal{A}'$ mit minimaler Anzahl starker Zusammenhangskomponenten und mit $L(\mathcal{A})\ \mathcal{BSR}\ L(\mathcal{A}')$ berechnet werden.*

Für den Beweis dieses Satzes, betrachten wir aber zuvor noch folgende zwei Möglichkeiten zur Konstruktion einer Gewinnstrategie für ein Erreichbarkeitsspiel:

Lemma 5.1. *Seien $\mathcal{A}, \mathcal{B}$ und $\mathcal{C}$ drei deterministische endliche Automaten und seien $A \in SCC(\mathcal{A})$, $B \in SCC(\mathcal{B})$, $C \in SCC(\mathcal{C})$ starke Zusammenhangskomponenten. Besitzt Eva für die Erreichbarkeitsspiele $\Gamma_{\mathcal{A},\mathcal{B}}$ und $\Gamma_{\mathcal{B},\mathcal{C}}$ an den Positionen (A, B) und (B, C) Gewinnstrategien, so lässt sich daraus eine Gewinnstrategie für das Spiel $\Gamma_{\mathcal{A},\mathcal{C}}$ an der Position (A, C) konstruieren.*

Beweis (Lemma 5.1)**.** Seien $\mathcal{A} = (Q_A, \Sigma_A, \delta_A, q_{0A}, F_A)$, $\mathcal{B} = (Q_B, \Sigma_B, \delta_B, q_{0B}, F_B)$ und $\mathcal{C} = (Q_C, \Sigma_C, \delta_C, q_{0C}, F_C)$. Seien zudem $\mathcal{A}' = (Q_A, \Sigma_A, \delta_A, q_A, F_A)$, $\mathcal{B}' = (Q_B, \Sigma_B, \delta_B, q_B, F_B)$ und $\mathcal{C}' = (Q_C, \Sigma_C, \delta_C, q_C, F_C)$ mit $q_A \in A$, $q_B \in B$ und $q_C \in C$ Unterautomaten von $\mathcal{A}$, $\mathcal{B}$ und $\mathcal{C}$, die in den starken Zusammenhangskomponenten A, B und C beginnen.

Besitzt *Eva* für die Position (A, B) in $\Gamma_{\mathcal{A},\mathcal{B}}$ eine Gewinnstrategie, dann gilt entsprechend der Charakterisierung des Bounded Streaming Repair Problems in Satz 3.2 für die Automaten $\mathcal{A}'$ und $\mathcal{B}'$, dass $L(\mathcal{A}') \leq_{\mathcal{BSR}} L(\mathcal{B}')$. Entsprechend folgt aus der Existenz einer Gewinnstrategie für *Eva* an der Position (B, C) im Erreichbarkeitsspiel $\Gamma_{\mathcal{B},\mathcal{C}}$, dass $L(\mathcal{B}') \leq_{\mathcal{BSR}} L(\mathcal{C}')$ und damit auch $L(\mathcal{A}') \leq_{\mathcal{BSR}} L(\mathcal{C}')$. Daraus folgt aber wiederum, dass *Eva* für das Erreichbarkeitsspiel $\Gamma_{\mathcal{A}',\mathcal{C}'}$ an der Position (A, C) eine Gewinnstrategie besitzt. Diese Gewinnstrategie ist nach Konstruktion der Erreichbarkeitsspiele auch eine Gewinnstrategie für die Position (A, C) im Spiel $\Gamma_{\mathcal{A},\mathcal{C}}$.

□

Lemma 5.2. *Seien $\mathcal{A}$ und $\mathcal{B}$ zwei deterministische endliche Automaten sowie $C \in SCC(\mathcal{A})$ und $D \in SCC(\mathcal{B})$ zwei starke Zusammenhangskomponenten und besitzt Eva für eine Position (C, D) im Erreichbarkeitsspiel $\Gamma_{\mathcal{A},\mathcal{B}}$ eine Gewinnstrategie, so besitzt Eva auch eine Gewinnstrategie für die Position (D, C).*

Beweis (Lemma 5.2)**.** Besitzt *Eva* für die Position (C, D) eine Gewinnstrategie, so existiert ein Zug $(C, D) \to (D', C)$ so, dass *Eva* auch für die Position (D', C) eine Gewinnstrategie besitzt. Insbesondere kann *Eva* jeden möglichen Zug $(D', C) \to (C', D')$ von *Adam* wieder durch einen Zug $(C', D') \to (D'', C')$ beantworten, so dass *Eva* für die Position (D'', C') wiederum eine Gewinnstrategie besitzt. Insbesondere existiert also für jede starke Zusammenhangskomponente C', die Nachfolger von C im Graph der starken Zusammenhangskomponenten $Dag(\mathcal{A})$ ist, ein starke Zusammenhangskomponente $D'' \in SCC(\mathcal{B})$ die Nachfolger von D im Graphen $Dag^*(\mathcal{B})$ ist, für die $L(\mathcal{A}|C') \subseteq L(\mathcal{B}|D'')$ gilt und für die *Eva* an der Position (D'', C') eine Gewinnstrategie im Erreichbarkeitsspiel $\Gamma_{\mathcal{A},\mathcal{B}}$ besitzt.

Von der Position (D, C) hat jeder Zug von *Adam* die Form (C', D) für eine starke Zusammenhangskomponente $C' \in SCC(\mathcal{A})$, für welche eine Kante $C \to C'$ in $Dag\mathcal{A}$ existiert. Daraus folgt aber, dass eine starke Zusammenhangskomponente $D'' \in SCC(\mathcal{B})$ mit $L(\mathcal{A}|C') \subseteq L(\mathcal{B}|D'')$ für die eine Kante $D \to D''$ im Graphen $Dag^*(\mathcal{B})$ existiert und für welche *Eva* an der Position (D'', C') im Erreichbarkeitsspiel $\Gamma_{\mathcal{A},\mathcal{B}}$ eine Gewinnstrategie besitzt. Insbesondere ist damit die Position (D'', C') ein direkter Nachfolger von (C', D) im Spielegraphen und *Eva* kann in ihrer Gewinnstrategie den Zug $(C', D) \to (D'', C)$ wählen. Somit besitzt *Eva* für alle Positionen der Form (C', D), für die C' ein Nachfolger von C im Graphen $Dag(\mathcal{A})$ ist, eine Gewinnstrategie und damit auch für die Position (D, C).

□

Mit Hilfe dieser Konstruktion, können wir nun den obigen Satz beweisen.

Beweis (Satz 5.2)**.** Die Berechnung des Automaten $\mathcal{A}'$ kann durch die Funktion MINIMIZEDAG in Algorithmus 5.3 erfolgen. Wir müssen folglich drei Dinge zeigen:

1. Für den durch MINIMIZEDAG berechneten Automaten $\mathcal{A}'$ gilt $L(\mathcal{A})\ \mathcal{BSR}\ L(\mathcal{A}')$.
2. Die Laufzeit von MINIMIZEDAG ist polynomiell in der Eingabelänge.
3. Die Anzahl der starken Zusammenhangskomponenten von $\mathcal{A}'$ ist minimal.

(1): Der Automat $\mathcal{A}'$ wird zunächst nach Algorithmus 5.1 aus dem Automaten $\mathcal{A}$ generiert. Für diesen Algorithmus hatten wir bereits gezeigt, dass $L(\mathcal{A})\ \mathcal{BSR}\ L(\mathcal{A}')$ gilt. Anschließend werden die Beliebigkeiten aus $\mathcal{A}'$ einzeln entfernt. In jedem dieser Schritte gilt für den neu generierten Automaten $\mathcal{A}''$, wie bereits festgestellt, $L(\mathcal{A}'')\ \mathcal{BSR}\ L(\mathcal{A}')$. Da $\mathcal{BSR}$ transitiv ist, folgt somit für den durch Algorithmus 5.1 generierten Automaten $\mathcal{A}'$, dass $L(\mathcal{A}')\ \mathcal{BSR}\ L(\mathcal{A})$ gilt.

(2): Für die Funktionen REMOVEEQUIVALENTSCCs aus Algorithmus 5.1 und FINDAMBIGUITY aus Algorithmus 5.2 haben wir bereits gezeigt, dass deren Laufzeit polynomiell in der Eingabelänge beschränkt ist. Auch für die Funktion REMOVEAMBIGUITY ist, entsprechend den vorhergehenden Ausführungen, leicht ersichtlich, dass diese sich effizient realisieren lässt. Die Berechnung des Spielegraphen $\Gamma_{\mathcal{A}}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ lässt sich ebenfalls in polynomieller Zeit realisieren (siehe auch [BPR11]). Es bleibt noch zu zeigen, dass die Anzahl der Ausführungen der While-Schleife in Algorithmus 5.3 polynomiell in der Größe des Eingabeautomaten beschränkt ist.

Innerhalb der While-Schleife wird der Automat $\mathcal{A}'$ durch die Funktion REMOVEAMBIGUITY verändert. Hierbei werden Kanten des Graphen $Dag(\mathcal{A}')$ entfernt und gegebenenfalls neue Kanten hinzugefügt. Es werden hierbei jedoch nur Kanten eingefügt, welche im reflexiv-transitiven Anschluss $Dag(\mathcal{A})$ bereits existieren. Die Kante, welche aus $Dag(\mathcal{A}')$ entfernt wird, besteht anschließend auch nicht mehr im reflexiv-transitiven Abschluss $Dag^*(\mathcal{A}')$. Die Anzahl der Kanten in $Dag^*(\mathcal{A}')$ nimmt somit bei jedem Schleifendurchlauf ab und die Anzahl der Schleifendurchläufe ist damit durch die Anzahl der Kanten des Graphen $Dag^*(\mathcal{A}')$ beschränkt.

(3): Nach Konstruktion besitzt der berechnete Automat weder Beliebigkeiten in der Gewinnstrategie von *Eva* für das entsprechende Erreichbarkeitsspiel, noch äquivalente starke Zusammenhangskomponenten. Wir zeigen hier durch Widerspruch, dass daraus auch folgt, dass die Anzahl der starken Zusammenhangskomponenten für den berechneten Automaten minimal ist.

Sei $\mathcal{A} = (Q_A, \Sigma_A, \delta_A, q_{0A}, F_A)$ ein deterministischer endlicher Automat ohne Beliebigkeit in der Gewinnstrategie von *Eva* für das Spiel $\Gamma_{\mathcal{A}}$ und ohne äquivalente starke Zusammenhangskomponenten, für welchen die Anzahl der starken Zusammenhangskomponenten nicht minimal ist. Dann existiert ein Automat $\mathcal{B} = (Q_B, \Sigma_B, \delta_B, q_{0B}, F_B)$ mit $L(\mathcal{A})\ \mathcal{BSR}\ L(\mathcal{B})$ und $|SCC(\mathcal{A})| < |SCC(\mathcal{B})|$. *Eva* besitzt also sowohl für das Erreichbarkeitsspiel $\Gamma_{\mathcal{A},\mathcal{B}}$ als auch für das Erreichbarkeitsspiel $\Gamma_{\mathcal{B},\mathcal{A}}$ an der Startposition $(\mathcal{C}(q_0A), \mathcal{C}(q_0B))$ bzw. $(\mathcal{C}(q_0B), \mathcal{C}(q_0A))$ eine Gewinnstrategie.

Nach Voraussetzung besitzt $\mathcal{A}$ keine äquivalenten starken Zusammenhangskomponenten. Daraus folgt, dass mindestens eine starke Zusammenhangskomponente $C \in SCC(\mathcal{A})$ existiert, so dass für alle starken Zusammenhangskomponenten $D \in SCC(\mathcal{B})$ gilt, dass *Eva* entweder für die Position (C, D) im Erreichbarkeitsspiel $\Gamma_{\mathcal{A},\mathcal{B}}$ oder für die Position (D, C)

im Erreichbarkeitsspiel $\Gamma_{\mathcal{B},\mathcal{A}}$ keine Gewinnstrategie besitzt. Ansonsten müssten nach dem Schubfachprinzip für eine starke Zusammenhangskomponente $D \in SCC(\mathcal{B})$ zwei starke Zusammenhangskomponenten C_i und $C_j \in SCC(\mathcal{A})$, $C_i \neq C_j$ existieren für die gilt, dass *Eva* sowohl für die Positionen (C_i, D) und (C_j, D) im Erreichbarkeitsspiel $\Gamma_{\mathcal{A},\mathcal{B}}$ als auch für die Positionen (D, C_i) und (D, C_j) im Erreichbarkeitsspiel $\Gamma_{\mathcal{B},\mathcal{A}}$ eine Gewinnstrategie besitzt. Daraus würde aber wiederum folgen, dass die Zusammenhangskomponenten C_j und C_k des Automaten $\mathcal{A}$ äquivalent sind, da sich aus diesen Gewinnstrategien entsprechend Lemma 5.1 wiederum eine Gewinnstrategien für *Eva* an den Positionen (C_i, C_j, Eva) und (C_j, C_i, Eva) im Erreichbarkeitsspiel $\Gamma_{\mathcal{A}}$ ableiten lassen würden.

Sei im Folgenden $C \in SCC(\mathcal{A})$ eine Zusammenhangskomponente, für die kein $D \in SCC(\mathcal{B})$ existiert, so dass *Eva* sowohl für die Position (C, D) im Erreichbarkeitsspiel $\Gamma_{\mathcal{A},\mathcal{B}}$ als auch für die Position (D, C) im Erreichbarkeitsspiel $\Gamma_{\mathcal{B},\mathcal{A}}$ eine Gewinnstrategie besitzt.

Betrachten wir das Erreichbarkeitsspiel $\Gamma_{\mathcal{A},\mathcal{B}}$, dann gibt es ein mit *Eva*s Gewinnstrategie für die Startposition $v_0 = (\mathcal{C}(q_{0A}), \mathcal{C}(q_{0B}))$ konformes Spiel $s = v_0 \dots v_{2n}$ mit $v_{2j} = (C_j, D_j)$ und $v_{2j+1} = (D_{j+1}, C_j)$ für $C_j \in SCC(\mathcal{A})$, $D_j \in SCC(\mathcal{B})$ und $C_j = C$ für ein $j \in \{0 \dots n\}$. Zu diesem Lauf existiert ein Pfad $\pi = C_0 \dots C_n$ im Graphen $Dag(\mathcal{A})$ und ein entsprechender Pfad $D_1 \dots D_{n+1}$ im Graphen $Dag^*(\mathcal{B})$. Für den Pfad $D_1 \dots D_{n+1}$ existiert im Graphen $Dag(\mathcal{B})$ ein Pfad $D'_0 \dots D'_m$, welcher insbesondere alle D_i für $i \in \{1 \dots n\}$ (in der entsprechenden Reihenfolge) enthält. Aus diesem Pfad $D'_0 \dots D'_m$ lässt sich wiederum ein Spiel $s' = v'_0 \dots v'_{2m}$, beginnend an der Startposition $v'_0 = (\mathcal{C}(q_{0B}), \mathcal{C}(q_{0A}))$ konstruieren, welches mit *Eva*s Gewinnstrategie für das Erreichbarkeitsspiel $\Gamma_{\mathcal{B},\mathcal{A}}$ konform ist und für welchen gilt $v'_{2j} = (D'_j, C'_j)$ und $v'_{2j+1} = (C_j + 1', D'_j)$ mit $C'_j \in SCC(\mathcal{A})$.

Mit Hilfe dieser beiden Spiele s und s' lässt sich nun wie folgt zeigen, dass *Eva* für das Spiel $\Gamma_{\mathcal{A}}$ an der Startposition $(\mathcal{C}(q_{0A}), \mathcal{C}(q_{0A}), Eva)$ eine nicht-triviale Gewinnstrategie besitzt:

Für jedes C_j im Pfad π existieren im Spiel s die Knoten $v_{2j} = (C_j, D_j)$ und $v_{2j+1} = (D_{j+1}, C_j)$. Daraus folgt, dass $L(\mathcal{A}|C_j) \subseteq L(\mathcal{B}|D_{j+1})$ gilt und da für jede starke Zusammenhangskomponente $D \in SCC(\mathcal{B})$ eine Kante $D \to D$ in $Dag^*(\mathcal{B})$ existiert, besitzt *Eva* somit auch im Erreichbarkeitsspiel $\Gamma_{\mathcal{A},\mathcal{B}}$ für die Position (C_j, D_{j+1}) eine Gewinnstrategie (sie kann von dort den Zug $(C_j, D_{j+1}) \to (D_{j+1}, C_j)$ wählen). Sei nun $D_{j+1} = D'_{k_j}$ für ein $k_j \in 0 \dots m$, dann enthält das Spiel s' die Knoten $v'_{2k_j} = (D'_{k_j}, C'_{k_j})$ und $v'_{2k_j+1} = (C'_{k_j+1}, D'_{k_j})$. Insbesondere folgt daraus, dass $L(\mathcal{B}|D_{k_j}) \subseteq L(\mathcal{A}|C'_{k_j+1})$ gilt und *Eva* an der Position (D'_{k_j}, C'_{k_j+1}) eine Gewinnstrategie besitzt.

Da *Eva* sowohl für die Position $(C_j, D_{j+1}) = (C_j, D'_{k_j})$ im Erreichbarkeitsspiel $\Gamma_{\mathcal{A},\mathcal{B}}$, als auch für die Position (D'_{k_j}, C'_{k_j+1}) im Erreichbarkeitsspiel $\Gamma_{\mathcal{B},\mathcal{A}}$ eine Gewinnstrategie besitzt, existiert nach Lemma 5.1 auch im Erreichbarkeitsspiel $\Gamma_{\mathcal{A}}$ an der Position (C_j, C_{k_j+1}, Eva) eine Gewinnstrategie für *Eva*. Nach Lemma 5.2 besitzt *Eva* somit auch eine Gewinnstrategie für die Position $(C_{k_j+1}, C_j, Adam)$.

Es existiert folglich eine Gewinnstrategie für *Eva* an der Startposition $(\mathcal{C}(q_{0A}), \mathcal{C}(q_{0A}), Eva)$ im Erreichbarkeitsspiel $\Gamma_{\mathcal{A}}$, bei der sich *Eva* wie folgt verhält. *Eva* wählt vom Startknoten $(\mathcal{C}(q_{0A}), \mathcal{C}(q_{0A}), Eva) = (C_0, C'_0, Eva)$ den Nachfolger $(C'_{k_0+1}, C_0, Adam)$ und jeden Zug

$(C'_{k_j+1}, C_j) \rightarrow (C_{j+1}, C'_{k_j+1})$ von *Adam* beantwortet *Eva* mit einem Zug $(C_{j+1}, C'_{k_j+1}) \rightarrow (C'_{k_{j+1}+1}, C_{j+1})$. Es existiert also, basierend auf dem Pfad $\pi = C_0 \dots C_n$, ein mit dieser Gewinnstrategie für *Eva* konformes Spiel $s'' = v''_0 \dots v''_{2n}$ mit $v''_{2j} = (C_j, C'_{k_j})$ und $v''_{2j+1} = (C'_{k_j+1}, C_j)$. Insbesondere gilt $C_j = C$ für ein $j \in \{1 \dots n\}$. Da für C kein $D \in SCC(\mathcal{B})$ existiert, für das *Eva* sowohl an der Position (C, D) in $\Gamma_{\mathcal{A},\mathcal{B}}$, als auch an der Position (D, C) in $\Gamma_{\mathcal{B},\mathcal{A}}$ eine Gewinnstrategie besitzt, gilt für dieses j: $C_j \neq C_{k_j+1}$. Somit ist s'' zu einer nicht-trivialen Gewinnstrategie von *Eva* für die Startposition $(\mathcal{C}(q_{0A}), \mathcal{C}(q_{0A}), Eva)$ konform und zeigt somit den Widerspruch zur Annahme.

□

6 Zusammenfassung

In dieser Arbeit wurde die Äquivalenz von regulären Sprachen mit einer beschränkten Anzahl von Fehlern analysiert. Hierfür wurde zunächst in Abschnitt 3.3 gezeigt, wie sich aus dem Bounded Repair Problem und dem Bounded Streaming Repair Problem in natürlicherweise die Äquivalenzrelationen $\mathcal{BR}$ und $\mathcal{BSR}$ auf Sprachen ergeben.

In Kapitel 4 wurden verschieden Sprachklassen analysiert. Als Basis wurden hierbei die algebraischen Eigenschaften Ideale, Filter und deren Boolesche Kombinationen analysiert und eine verbanstheoretische Charakteriesierung der entsprechenden Sprachklassen vorgestellt. Dadurch konnten für rechtsseitige Ideale (Proposition 4.1), rechtsseitige und beidseitige Filter (Korollar 4.1 und 4.2) sowie Boolesche Kombinationen von rechtsseitigen Idealen (Satz 4.2) Automatenmodelle angegeben werden, welche die entsprechenden Sprachklassen vollständig beschreiben. Diese bieten eine Möglichkeit zur Klassifizierung von regulären Sprachen, die für ω-Sprachen ihre Entsprechung in der Cantor-Topologie findet. Diese Ergebnisse wurden bereits in einer Arbeit mit Manfred Kufleitner und Alexander Lauser veröffentlicht [JKL12]. Für die Klasse der stark zusammenhängenden Pfadautomaten konnte zudem gezeigt werden, dass das Inklusionsproblem im allgemeinen Fall PSPACE hart ist, aber bei einer deterministischen Übergangsfunktion in PTIME liegt.

In Kapitel 5 konnten mit Hilfe dieser Ergebnisse in Satz 5.1 zeigen, dass das Minimierungsproblem mit einer beschränkten Anzahl von Fehlern im „Nonstreaming“ Fall PSPACE hart ist. Hierzu wurde das Inklusionsproblem für stark zusammenhängende Pfadautomaten auf dieses Problem reduziert. Zudem konnte gezeigt werden, dass jede $\mathcal{BR}$-Klasse eine unter Infix abgeschlossene Sprache enthält (Korollar 5.1).

Für den „Streaming“ Fall konnten wir zeigen, dass jede $\mathcal{BSR}$-Klasse eine unter Präfix abgeschlossene Sprache enthält (Korollar 5.2). Zur Minimierung betrachteten wir Erreichbarkeitsspiele, die zur Entscheidung des Bounded Streaming Repair Problems herangezogen werden können. Für die Analyse eines deterministischen Automaten $\mathcal{A}$ konnten wir mit dieser Technik ein Erreichbarkeitsspiel $\Gamma_{\mathcal{A}}$ konstruieren, welches die notwendigen Informationen über die Eigenschaften des Automaten enthält. Es wurde ein Algorithmus entwickelt, welcher alle nicht-trivialen Gewinnstrategien für *Eva* im Spiel $\Gamma_{\mathcal{A}}$ eliminiert und äquivalente Zusammenhangskomponenten des Automaten zusammenlegt. Durch die Analyse verschiedener Eigenschaften des Erreichbarkeitsspiels und der Gewinnstrategien, konnten wir zeigen, dass die Anzahl der starken Zusammenhangskomponenten des so konstruierten Automaten minimal ist (Lemma 5.1 und 5.2, Satz 5.1).

Ausblick

Die Minimierung von Automaten mit einer beschränkten Anzahl von Fehlern wurde im Rahmen dieser Arbeit für reguläre Sprachen analysiert. Benedikt, Puppis und Riveros haben gezeigt, dass das Bounded Repair Problem auch für ω-Sprachen durch die Analyse von Büchi-Automaten entschieden werden kann. Hierbei konnten sie die Charakterisierung des Bounded Repair Problems, mit einigen Modifikationen, auch auf Büchi-Automaten übertragen. Offen blieb dabei, ob sich auch die Ergebnisse des Bounded Streaming Repair Problems ebenso auf ω-Sprachen übertragen lassen. Basierend auf den bisherigen Ergebnissen stellt sich die Frage, ob sich für Büchi-Automaten und alternative Automatenmodelle für ω-Sprachen in ähnlicher Weise minimale Automaten mit einer beschränkten Anzahl von Fehlern finden lassen.

Des Weiteren können die Eigenschaften der in Kapitel 4 betrachteten Sprachklassen und Automatenmodelle für weitere Anwendungsfälle betrachtet werden. In [JKL12] wurden als Anwendungsfall beispielsweise Zweiwege-Automaten und die Sprachen der Varietät $\mathcal{DA}$ untersucht.

Literaturverzeichnis

[AF06] S. V. Avgustinovich, A. E. Frid. Canonical Decomposition of a Regular Factorial Language. In D. Grigoriev, J. Harrison, E. A. Hirsch, editors, *CSR*, volume 3967 of *Lecture Notes in Computer Science*, pp. 18–22. Springer, 2006. (Zitiert auf Seite 25)

[AH74] A. V. Aho, J. E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1974. (Zitiert auf Seite 29)

[BPR11] M. Benedikt, G. Puppis, C. Riveros. Regular repair of specifications. *26th Annual IEEE Symposium on Logic in Computer Science (LICS)*, 2011. (Zitiert auf den Seiten 5, 10, 13, 14, 16, 17, 27, 28 und 42)

[Ham50] R. Hamming. Error Detecting and Error Correcting Codes. *Bell System Techincal Journal*, 29:147–160, 1950. (Zitiert auf Seite 10)

[JKL12] F. Jahn, M. Kufleitner, A. Lauser. Regular Ideal Languages and Their Boolean Combinations. In N. Moreira, R. Reis, editors, *CIAA*, volume 7381 of *Lecture Notes in Computer Science*, pp. 205–216. Springer, 2012. (Zitiert auf den Seiten 21, 45 und 46)

[MSS92] D. E. Muller, A. Saoudi, P. E. Schupp. Alternating Automata, the Weak Monadic Theory of Trees and its Complexity. *Theor. Comput. Sci.*, 97(2):233–244, 1992. (Zitiert auf Seite 25)

[PP65] A. Paz, B. Peleg. Ultimate-Definite and Symmetric-Definite Events and Automata. *J. ACM*, 12(3):399–410, 1965. (Zitiert auf Seite 24)

[PP04] D. Perrin, J.-É. Pin. *Infinite words*, volume 141 of *Pure and Applied Mathematics*. Elsevier, Amsterdam, 2004. (Zitiert auf Seite 21)

[SK00] D. Sankoff, J. B. Kruskal. *Time warps, string edits, and macromolecules*. Cambridge University Press, Cambridge, England, 2000. (Zitiert auf Seite 10)

[SW74] L. Staiger, K. W. Wagner. Automatentheoretische und automatenfreie Charakterisierungen topologischer Klassen regulärer Folgenmengen. *Elektronische Informationsverarbeitung und Kybernetik*, 10(7):379–392, 1974. (Zitiert auf Seite 25)

[Tho90] W. Thomas. Automata on Infinite Objects. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pp. 133–192. 1990. (Zitiert auf Seite 21)

[Wag74] R. A. Wagner. Order-n correction for regular languages. *Commun. ACM*, 17(5):265–268, 1974. (Zitiert auf Seite 10)

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Franz G. Jahn)