Inclusion of Pattern Languages and Related Problems

Dominik D. Freydenberger

Goethe-Universität, Frankfurt am Main
freydenberger@em.uni-frankfurt.de

Abstract: Patternsprachen sind ein einfacher und eleganter Mechanismus zur Beschreibung von Sprachen, deren Wörter über Wiederholungen definiert sind. Trotz dieser Einfachheit sind viele der kanonischen Fragestellungen für Patternsprachen überraschend schwer zu lösen. Die vorliegende Arbeit befasst sich mit verschiedenen Aspekten des Inklusionsproblems für Patternsprachen. Neben Beweisen zur Unentscheidbarkeit dieses Problems, selbst für verschiedene stark eingeschränkte Unterklassen, werden die Resultate auf *regex*, eine in modernen Programmiersprachen weit verbreitete Erweiterung der regulären Ausdrücke übertragen. Ein weiterer Schwerpunkt der Untersuchungen sind die Existenz und Berechnung deskriptiver Pattern, welche inklusionsminimale Verallgemeinerungen beliebiger Sprachen durch Patternsprachen darstellen.

1 Pattern und ihre Sprachen

Pattern, d. h. endliche Wörter aus Variablen und Terminalsymbolen, stellen eine kompakte, elegante und natürliche Methode dar, gewisse Sprachen mit Wörtern mit Wiederholungen zu repräsentieren. Ein Pattern erzeugt ein Wort durch eine Substitution, die alle Variablen im Pattern durch beliebige endliche Wörter über einem festen Terminalalphabet ersetzt. Die *Patternsprache* eines Pattern ist somit die Menge aller Wörter, die durch die Substitution aus dem Pattern gebildet werden können; etwas formaler ausgedrückt ist eine Patternsprache also die Menge aller Bilder des Pattern unter beliebigen terminalerhaltenden Homomorphismen. Wenn wir beispielsweise das Pattern $\alpha{:=}x_1\,\mathtt{a}\,x_2\,\mathtt{b}\,x_1$ (mit Variablen x_1, x_2 und Terminalsymbolen $\mathtt{a}, \mathtt{b}$) betrachten, liegen folglich (unter anderem) die Wörter $w_1{:=}\mathtt{aabbba}$, $w_2{:=}\mathtt{abababab}$ und $w_3{:=}\mathtt{aaabaa}$ in der von α erzeugten Patternsprache $L(\alpha)$, wogegen die Beispielwörter $w_4{:=}\mathtt{ba}$, $w_5{:=}\mathtt{babbba}$ und $w_6{:=}\mathtt{abba}$ nicht von α erzeugt werden können.

Da jedes Vorkommen einer mehrfach auftretenden Variablen durch eine (feste) Substitution stets gleich ersetzt werden muss, können Patternsprachen als eine einfache Formalisierung des Wiederholungsoperators verstanden werden. Die Untersuchung von unvermeidbaren Pattern in Wörtern lässt sich bis zu [Thu06] zurückverfolgen und zählt mittlerweile zu den etablierten Teilgebieten der Wortkombinatorik (siehe [Cas02]), während die explizite Verwendung von Pattern als Sprachgeneratoren von Angluin in [Ang80] eingeführt wurden. In Angluins Definition ist es nicht erlaubt, Variablen löschend zu ersetzen, daher wird diese Klasse von Patternsprachen in der Literatur im Allgemeinen als

NE-Patternsprachen bezeichnet („NE“ steht in diesem Fall für „non-erasing“). Die NE-Patternsprache eines Patterns α über einem Alphabet Σ bezeichnen wir mit $L_{\mathrm{NE},\Sigma}(\alpha)$.

Ebenfalls häufig betrachtet werden die von [Shi82] eingeführten sogenannten *E-Patternsprachen* („E“ wie „erasing“ oder „extended“); bei dieser Sprachklasse ist die löschende Substitution gestattet. (Analog zu $L_{\mathrm{NE},\Sigma}(\alpha)$ verwenden wir die Bezeichnung $L_{\mathrm{E},\Sigma}(\alpha)$ für die E-Patternsprache von α.) Im obigen Beispiel ist somit das Wort w_3 zwar in der E-Patternsprache $L_{\mathrm{E},\Sigma}(\alpha)$, jedoch nicht in der NE-Patternsprache $L_{\mathrm{NE},\Sigma}(\alpha)$ enthalten. .

Trotz des – scheinbar kleinen – definitorischen Unterschieds zwischen E- und NE-Patternsprachen besitzen die beiden Sprachklassen sehr unterschiedliche Eigenschaften. Beispielsweise lässt sich das *Äquivalenzproblem* (d. h. die Frage, ob zwei Pattern die gleiche Sprache erzeugen) für *NE*-Patternsprachen trivial in Polynomialzeit entscheiden, während die Entscheidbarkeit des Äquivalenzproblems für *E*-Patternsprachen ein schweres und seit mehr als 25 Jahren offenes Problem darstellt (siehe [OU97, Rei07]).

Beide Sprachklassen haben ihren Ursprung in der Induktiven Inferenz, einem Teilgebiet der (mathematischen) Lerntheorie. Ziel dieser Betrachtungen war es, zu gegebenen Mengen von Wörtern effektiv Pattern zu finden, die all diese Wörter beschreiben. Inzwischen wurden Patternsprachen auch in der Theorie der formalen Sprachen ausgiebig untersucht. Wegen ihrer einfachen Definition kommen Pattern und ihre Sprachen in einer Vielzahl anderer Gebiete der Informatik und der diskreten Mathematik vor: Die in modernen Programmiersprachen verwendeten *erweiterten regulären Ausdrücke* (auch *regex* genannt, siehe Abschnitt 3), *Wortgleichungssysteme*, verschiedene *Datenbankanfragesprachen* – kurz, nahezu alle Modelle, die einen Wiederholungsoperator verwenden – können als Erweiterung oder Anwendung von Patternsprachen verstanden werden.

Aufgrund ihrer einfachen Definition ließe sich vermuten, dass die meisten interessanten Eigenschaften einer Patternsprache direkt an dem sie erzeugenden Pattern zu erkennen sind. Insbesondere sollte sich daher eigentlich leicht entscheiden lassen, ob zwei Pattern die gleiche oder vergleichbare Sprachen erzeugen (d. h. ob eine Äquivalenz- oder Inklusionsbeziehung vorliegt), und ob ein Wort in der Sprache eines Patterns ist (das sogenannte *Wortproblem*). Tatsächlich ist diese Vermutung aber in den meisten Fällen ein Trugschluss – beispielsweise ist das Wortproblem NP-vollständig und das Inklusionsproblem im Allgemeinen unentscheidbar.

Diese unteren Schranken für Probleme zu Patternsprachen gelten selbstverständlich auch unmittelbar für alle mächtigeren Modelle, die Patternsprachen erweitern. Häufig folgt in der theoretischen Informatik auf das Feststellen einer unteren Schranke für ein Problem die Frage nach geeigneten Einschränkungen, mit denen diese Schranke überwunden werden kann. Hierbei bieten sich Patternsprachen als „Versuchsgelände“ für eine Vielzahl dieser Einschränkungen an, denn eine Einschränkung, die für Patternsprachen nicht zu besseren unteren Schranken führt, kann auch bei den stärkeren Modellen nicht greifen.

Jede Erkenntnis über die Schwierigkeiten im Umgang mit Patternsprachen lässt sich daher unmittelbar auf die mächtigeren Klassen übertragen. Gleichzeitig lassen sich die für Patternsprachen entwickelten Beweise in mächtigeren Modellen oft erweitern, so dass deutlich stärkere Resultate erzielt werden können. Auf diese Art stellen Patternsprachen eine Art „Werkzeugkasten“ für andere Modelle zur Verfügung. Zwei Beispiele für die Anwen-

dung dieses Prinzips werden in den Abschnitten 3 und 7 vorgestellt.

Die hier zusammengefasste Dissertation [Fre11b] beschäftigt sich hauptsächlich mit verschiedenen Aspekten des Inklusionsproblems. Den eigentlichen Hauptteil der Arbeit bilden die Kapitel 3 bis 7, die sich grob in zwei Teile unterteilen lassen.

Der erste Teil, der aus den Kapiteln 3 und 4 besteht, befasst sich direkt mit dem Inklusionsproblem für Patternsprachen und wendet eine in diesem Kontext entwickelte Technik auf eine in der Praxis weit verbreitete Erweiterung der regulären Ausdrücke an.

Der zweite Teil, bestehend aus den Kapiteln 5 bis 7, untersucht Fragestellungen zu sogenannten *deskriptive Pattern*, die aufgrund ihrer Definition eng mit dem Inklusionsproblem verwandt sind.

Im Folgenden wird der Inhalt der einzelnen Kapitel genauer vorgestellt und jeweils explizit erwähnt, in welcher Form die genannten Resultate bereits veröffentlicht wurden. Aus Platzgründen wird auf die meisten Definitionen verzichtet.

2 Inklusion von Patternsprachen (Kapitel 3)

Die Entscheidbarkeit des Inklusionsproblems für Patternsprachen wurde bereits bei der Einführung der NE-Patternsprachen durch [Ang80] als offenes Problem benannt. Sowohl der NE- als auch der E-Fall dieses Problems erwiesen sich als überraschend schwer zu lösen, bis schließlich in [JSSY95] der Beweis für die Unentscheidbarkeit beider Fälle gelang.

Allerdings basiert der in [JSSY95] angegebene Beweis auf der Annahme, dass die darin verwendeten Patternsprachen auf Terminalalphabeten von unbeschränkter Größe definiert werden können. Im Gegensatz dazu verwenden aber die meisten Anwendungen von Patternsprachen Alphabete von beschränkter Größe. Trotz des wegweisenden Resultats von [JSSY95] blieb die Entscheidbarkeit des Inklusionsproblems über festen Alphabeten offen. Das erste Hauptresultat der vorliegenden Arbeit beantwortet diese Frage negativ:

Theorem 3.3 *Sei Σ ein endliches Alphabet, $|\Sigma| \geq 2$. Das Inklusionsproblem für E-Patternsprachen über Σ ist unentscheidbar.*

Mittels einer Reduktion aus [JSSY95] folgt hieraus außerdem die Unentscheidbarkeit des Inklusionsproblems für NE-Patternsprachen über endlichen Alphabeten mit mindestens vier Buchstaben. Theorem 3.3 beantwortet die entsprechenden offenen Fragen von [Rei06] und [Sal06] und wurde bereits kurz nach seiner Veröffentlichung von [BHLW10] in der Datenbanktheorie angewendet.

Im restlichen Teil von Kapitel 3 wird die Entscheidbarkeit der Inklusion für stärker eingeschränkte Klassen von Patternsprachen untersucht. Sowohl der ursprüngliche Beweis von Jiang et al., als auch der darauf aufbauende Beweis von Theorem 3.3 setzen voraus, dass die verwendeten Pattern unbeschränkt viele verschiedene Variablen enthalten dürfen.

Daher ist es naheliegend, als weitere Einschränkung die Anzahl der in den Pattern vorkommenden Variablen zu betrachten. Eine Verfeinerung des Beweises von Theorem 3.3

zeigt, dass das Inklusionsproblem für E-Patternsprachen mit einer beschränkten (wenn auch vergleichsweise hohen) Anzahl von Variablen weiterhin unentscheidbar ist:

Theorem 3.10. *Ist $|\Sigma| = 2$, so ist das Inklusionsproblem $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$ für Pattern mit beschränkter Variablenanzahl unentscheidbar, wenn*

1. *α auf 3 und β auf 2854 Variablen beschränkt ist, oder*
2. *α auf 2 und β auf 2860 Variablen beschränkt ist.*

Um für weiter reduzierte Variablenanzahlen interessante Resultate unterhalb der Unentscheidbarkeit zu finden, wird (analog zu einer Technik aus der Untersuchung kleiner Turingmaschinen) außerdem ein sehr einfaches Berechnungsmodell auf die Inklusion von Patternsprachen reduziert. Hierzu wird die Iteration der *Collatzfunktion $\mathcal{C}$* (siehe [Lag09a, Lag09b]) betrachtet[1]. Die *Collatzvermutung* besagt, dass die wiederholte Iteration von $\mathcal{C}$ für alle Startwerte zum *trivialen Zyklus* 4, 2, 1 führt.

Hier stellt sich heraus, dass bereits eine verhältnismäßig geringe Anzahl von Variablen genügt, um die Iteration der Collatzfunktion in Patternsprachen zu codieren:

Theorem 3.11. *Sei $|\Sigma| = 2$. Jeder Algorithmus, der die Inklusion $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$ für Pattern mit beschränkter Variablenanzahl entscheidet, wobei α auf 2 und β auf 74 Variablen beschränkt ist, kann effektiv in einen Algorithmus konvertiert werden, der für jedes $n \geq 1$ entscheidet, ob die Iteration von $\mathcal{C}$ mit Startwert n zum trivialen Zyklus führt.*

Dieses Resultat lässt sich leicht zu dem folgenden mächtigeren Theorem erweitern:

Theorem 3.12. *Sei $|\Sigma| = 2$. Jeder Algorithmus, der die Inklusion $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$ für Pattern mit beschränkter Variablenanzahl entscheidet, wobei α auf 4 und β auf 80 Variablen beschränkt ist, kann effektiv in einen Algorithmus konvertiert werden, der die Existenz nicht-trivialer Zyklen von $\mathcal{C}$ entscheidet.*

Ein entsprechender Entscheidungsalgorithmus könnte also in endlicher Zeit entweder die Collatzvermutung widerlegen, oder aber die Nichtexistenz einer von zwei möglichen Klassen von Gegenbeispielen beweisen.

Auch wenn aus diesen Resultaten keine Unentscheidbarkeit der Inklusionsprobleme von Patternsprachen mit den entsprechenden Variablenanzahlen folgt, zeigen die Theoreme 3.11 und 3.12, dass die entsprechenden Probleme wenn nicht unentscheidbar, so doch zumindest schwer zu lösen sind. Alle hier aufgeführten Resultate lassen sich mit ähnlichen Variablenanzahlen auf größere (endliche) Alphabete und auf NE-Patternsprachen übertragen.

Die Inhalte dieses Kapitels wurden in den Arbeiten [FR10a] (zuerst erschienen als [FR08]) und [BF10] veröffentlicht.

3 Erweiterte reguläre Ausdrücke (Kapitel 4)

Reguläre Ausdrücke zählen zu den am weitesten verbreiteten Beschreibungsmechanismen und werden sowohl in der theoretischen, als auch in der praktischen Informatik auf ver-

[1] Die Funktion $\mathcal{C}$ ist definiert durch $\mathcal{C}(n) := 3n + 1$ für ungerade n, und $\mathcal{C}(n) := \frac{1}{2}n$ für gerade n.

schiedenste Arten angewendet. Allerdings haben sich im Lauf der Jahrzehnte in Theorie und Anwendung zwei unterschiedliche Interpretationen dieses Konzepts entwickelt.

Während die Theorie weitestgehend der klassischen Definition folgt und reguläre Ausdrücke betrachtet, die genau die Klasse der regulären Sprachen beschreiben, erlauben die meisten modernen Implementierungen von regulären Ausdrücken (so zum Beispiel in Perl, Java, C#) die Spezifikation von Wiederholungen mittels *Variablen* (oder *Rückreferenzen*). Die daraus resultierenden *erweiterten regulären Ausdrücke*, auch *regex* genannt, können dank dieser Wiederholungen auch nichtreguläre Sprachen beschreiben.

Beispielsweise erzeugt der erweiterte reguläre Ausdruck $((\mathtt{a} \mid \mathtt{b})^*)\%x\, x$ die nichtreguläre Sprache $\{ww \mid w \in \{\mathtt{a},\mathtt{b}\}^*\}$. Hierbei kann der Teilausdruck $((\mathtt{a} \mid \mathtt{b})^*)\%x$ ein beliebiges Wort $w \in \{\mathtt{a},\mathtt{b}\}^*$ erzeugen, während zudem dieses Wort w in der Variablen x gespeichert wird. Weitere Vorkommen von x erzeugen exakt das gleiche Wort w.

Andererseits führt die Verwendung von Variablen nicht zwangsläufig zu Nichtregularität; beispielsweise erzeugt (für jedes $n \geq 1$) der Ausdruck

$$\alpha_n := (\underbrace{(\mathtt{a} \mid \mathtt{b}) \ldots (\mathtt{a} \mid \mathtt{b})}_{n \text{ mal } (\mathtt{a} \mid \mathtt{b})})\%x\, x$$

die endliche (und daher reguläre) Sprache aller Wörter $ww \in \{\mathtt{a},\mathtt{b}\}^*$, die die Länge $2n$ haben. Hierbei fällt auf, dass *klassische reguläre Ausdrücke* (d. h. Ausdrücke, die keine Variablen enthalten) für diese Sprachen exponentiell länger sind als die Ausdrücke α_n.

Kapitel 4 befasst sich hauptsächlich mit den folgenden zwei Fragen: Erstens, können erweiterte reguläre Ausdrücke – in Hinsicht auf ihre Länge oder auf ihre Variablenanzahl – effektiv minimiert werden? Und zweitens, um wie viel kompakter ist die Beschreibung von regulären Sprachen durch erweiterte reguläre Ausdrücke im Vergleich zu klassischen regulären Ausdrücken?

Die Klasse der Patternsprachen ist eine Teilklasse der von erweiterten regulären Ausdrücken erzeugten Sprachen, da sich die erzeugenden Pattern ohne großen Aufwand direkt in die entsprechenden Ausdrücke konvertieren lassen. Allerdings erhöht die Verwendung des Alternierungsoperators $\mid$ die Ausdruckskraft so sehr, dass deutlich schärfere Nichtentscheidbarkeitsresultate als die in Kapitel 3 enthaltenen Resultate zur Inklusion für Patternsprachen erzielt werden können.

Die Konstruktion zum Beweis von Theorem 3.10 lässt sich mit signifikantem Zusatzaufwand zu einer mächtigeren Konstruktion für erweiterte reguläre Ausdrücke umbauen (Theorem 4.14), mit deren Hilfe in Theorem 4.15 mehrere Nichtentscheidbarkeitsresultate gewonnen werden:

Theorem 4.15. *Für erweiterte reguläre Ausdrücke ist die Allspracheneigenschaft nicht entscheidbar; Regularität sowie Endlichkeit des Komplements sind weder semi-entscheidbar, noch co-semi-entscheidbar.*

Mittels dieser Resultate können die beiden weiter oben genannten Fragen beantwortet werden: Erweiterte reguläre Ausdrücke können nicht effektiv minimiert werden, und der Größenunterschied zwischen erweiterten und klassischen regulären Ausdrücken ist durch keine berechenbare Funktion beschränkt.

Besondere Erwähnung verdient hierbei die Tatsache, dass die genannten negativen Eigenschaften bereits bei erweiterten regulären Ausdrücken mit einer einzigen Variable gelten.

Als praktische Konsequenz lässt sich feststellen, dass selbst die Verwendung einer einzigen Variable zwar deutlich kompaktere Ausdrücke erlauben kann (was sich natürlich auch positiv auf die Geschwindigkeit des Matchens der Ausdrücke auswirkt), dass aber andererseits dieser Vorteil aufgrund der Unmöglichkeit einer effektiven (oder gar effizienten) Minimierung nicht generell nutzbar ist.

Es ist zu hoffen, dass geeignete Teilklassen der erweiterten regulären Ausdrücke gefunden werden können, die diese enormen Kompressionsmöglichkeiten möglichst umfassend ausnutzen und sich trotzdem effektiv (und vorzugsweise auch effizient) aus klassischen regulären Ausdrücken berechnen lassen.

Die Inhalte dieses Kapitels wurden in der Arbeit [Fre12] (zuerst erschienen als [Fre11a]) veröffentlicht.

4 Existenz deskriptiver Pattern (Kapitel 5)

Ein Pattern α ist *konsistent* mit einer Menge $S \subseteq \Sigma^*$, wenn jedes Wort von S in der von α erzeugten Sprache $L(\alpha)$ enthalten ist[2], d. h. wenn $L(\alpha) \supseteq S$ gilt. So sind beispielsweise die Pattern $\alpha_0{:=}x$, $\alpha_1{:=}xyxyx$ und $\alpha_2{:=}x\,\mathtt{a}\,\mathtt{b}\,y$ konsistent mit der Menge $S_0{:=}\{\mathtt{ababa}, \mathtt{ababbababbab}, \mathtt{babab}\}$. Konsistente Pattern liefern also eine kompakte und leicht verständliche Darstellung von Gemeinsamkeiten der Wörter einer Menge.

Wie das oben stehende Beispiel zeigt, gibt es zu einer Menge von Wörtern im Allgemeinen eine Vielzahl von konsistenten Pattern, die intuitiv eine sehr unterschiedliche Güte haben können; beispielsweise ist das Pattern α_0 mit jeder Sprache konsistent und dürfte daher für nahezu alle Anwendungen als eine triviale und uninteressante Approximation gelten.

Es ist daher vorteilhaft, konsistente Pattern hoher Qualität formal zu fassen. Ein in der Literatur häufig verwendetes Qualitätsmaß ist das Konzept der *deskriptiven* Pattern für eine Menge S. Ein Pattern δ dessen Sprache $L(\delta)$ in einer gegebenen Klasse P von Patternsprachen liegt, ist P-deskriptiv für eine Sprache S, wenn δ mit S konsistent ist und außerdem $L(\delta) \supset L(\gamma) \supseteq S$ für kein Pattern γ mit $L(\gamma) \in P$ gilt.

Anschaulich formuliert kann ein P-deskriptives Pattern als Erzeuger einer kleinsten innerhalb der Klasse P möglichen Generalisierung der Zielsprache S verstanden werden. Die vorliegende Arbeit betrachtet hierbei vor allem die Klassen der NE-Patternsprachen, der E-Patternsprachen, und die Klasse der *terminalfreien E-Patternsprachen* (also derjenigen E-Patternsprachen, die von Pattern erzeugt werden, die keine Terminalsymbole enthalten). In den ersten beiden Fällen sprechen wir gewöhnlich von *NE-deskriptiven* beziehungsweise *E-deskriptiven* Pattern.

Die wahrscheinlich naheliegendste Frage bei der Suche nach deskriptiven Pattern für beliebige Sprachen ist, ob zu jeder Sprache mindestens ein deskriptives Pattern existiert. Für

[2] Hierbei ist natürlich von Fall zu Fall festzulegen, ob E- oder NE-Patternsprachen betrachtet werden.

NE-Patternsprachen wurde diese Frage bereits von [Ang80] positiv beantwortet. Ebenso wurde von Jiang et al. [JKS+94] bewiesen, dass alle endlichen Sprachen ein E-deskriptives Pattern besitzen, während der Fall von E-deskriptiven Pattern für unendliche Sprachen offen blieb. Das Hauptresultat des Kapitels beantwortet diese Frage negativ:

Theorem 5.18. *Zu jedem Alphabet Σ mit $|\Sigma| \geq 2$ existiert eine unendliche Sprache $L_\Sigma \subset \Sigma^*$, für die kein Pattern E-deskriptiv ist.*

Um die durch die Nichtentscheidbarkeit des Inklusionsproblems entstehenden Schwierigkeiten zu umgehen, verwendet der Beweis von Theorem 5.18 terminalfreie E-Patternsprachen, da für diese Unterklasse die Inklusion durch ein einfaches syntaktisches Kriterium charakterisiert wird.

Die meisten Inhalte dieses Kapitels wurden in der Arbeit [FR10b] (zuerst erschienen als [FR09]) veröffentlicht.

5 Deskriptive Generalisierung (Kapitel 6)

Im Gegensatz zu Kapitel 5, in dem die Existenz deskriptiver Pattern im Vordergrund steht, befasst sich dieses Kapitel mit der Frage nach ihrer effektiven Auffindbarkeit.

Hierzu wird das Konzept der *deskriptiven Generalisierung (anhand von positiven Daten)* eingeführt (kurz: das DG-Modell), das an Golds klassisches Modell (siehe [Gol67]) der *Identifikation von Sprachen im Limes (anhand von positiven Daten)*, das LIM-TEXT-Modell angelehnt ist. Das LIM-TEXT-Modell wurde in der einschlägigen Literatur intensiv untersucht, ein recht aktueller Übersichtsartikel ist [NS08].

Anschaulich (und vereinfacht) ausgedrückt untersuchen wir die Existenz von Lernstrategien, die Positivbeispiele der zu lernenden Sprache L wortweise einlesen und, wann immer ein bisher ungesehenes Wort eingelesen wird, ein Pattern als *Hypothese* ausgeben.

Ist P eine Klasse von Patternsprachen, so ist eine Klasse $\mathcal{L}$ von Sprachen *P-deskriptiv generalisierbar*, wenn eine berechenbare *Generalisierungsstrategie* S existiert, so dass für jede Sprache $L \in \mathcal{L}$ die Folge der von S ausgegeben Hypothesen gegen ein Pattern δ konvergiert, das P-deskriptiv für L ist.

Es muss also keine exakte Repräsentation der Zielsprachen erlernt werden, sondern nur eine Approximation. Hierbei stellt sich heraus, dass die Korrespondenz zwischen zu generalisierenden Sprachen und korrekten Hypothesen im Allgemeinen schwächer ist als bei den üblicherweise in der Literatur betrachteten Lernmodellen: Ein Pattern δ kann gleichzeitig deskriptives Pattern (und korrekte Hypothese) für zwei unvergleichbare Sprachen L_1, L_2 sein, und eine Sprache L kann durch zwei unterschiedliche deskriptive Pattern δ_1 und δ_2 generalisiert werden.

Um Probleme mit der Unentscheidbarkeit der Inklusion zu vermeiden, wird das Modell anhand der Klasse $\mathrm{ePAT}_{\mathrm{tf}}$ der teminalfreien E-Patternsprachen untersucht. Im Mittelpunkt der Untersuchung steht hierbei das Konvergenzverhalten der *kanonischen Strategie* Canon. Diese berechnet zu jeder Menge von Positivbeispielen aus der Zielsprache ein de-

skriptives Pattern. Die Klasse der Sprachen, auf denen Canon korrekt konvergiert, lässt sich wie folgt charakterisieren:

Theorem 6.26. *Sei Σ ein Alphabet mit $|\Sigma| \geq 2$. Für jede Sprache $L \subseteq \Sigma^*$, und jede Aufzählung $t : \mathbb{N} \to L$ von Positivbeispielen von L konvergiert* Canon *korrekt auf t genau dann, wenn L ein telling set besitzt.*

Ein *telling set* einer Sprache L ist hierbei eine endliche Teilmenge $T \subseteq L$, so dass ein terminalfreies Pattern δ existiert, dass sowohl für T als auch für L $\mathrm{ePAT}_{\mathrm{tf}}$-deskriptiv ist. Mittels dieser Charakterisierung lässt sich eine große deskriptiv generalisierbare Klasse von Sprachen definieren, wodurch tiefere Einsichten zur Stärke des DG-Modells gewonnen und ein Vergleich zum $\mathrm{LIM\text{-}TEXT}$-Modell ermöglicht werden.

Die Inhalte dieses Kapitels wurden in der Arbeit [FR12] (zuerst erschienen als [FR10c]) veröffentlicht.

6 Über eine Vermutung zu deskriptiven Pattern (Kapitel 7)

Dieses Kapitel befasst sich mit einer Vermutung, die der Autor beim Versuch aufstellte, den Beweis von Theorem 5.18 zu einer Charakterisierung derjenigen Sprachen zu erweitern, für die kein Pattern in Bezug auf die Klasse der terminalfreien E-Pattersprachen deskriptiv ist.

Dazu wird ein größeres technisches Instrumentarium eingeführt und anschließend zur Konstruktion von Gegenbeispielen zu dieser Vermutung verwendet. Darüber hinaus werden verschiedene Phänomene diskutiert, die die Schwierigkeit einer Charakterisierung der genannten Sprachen verdeutlichen.

Als Fazit dieses Kapitels lässt sich feststellen, dass die Nichtexistenz deskriptiver Pattern selbst in Bezug auf die Klasse der terminalfreien E-Patternsprachen komplex und wahrscheinlich nur schwer zu charakterisieren ist.

7 Zusammenfassung und neuere Resultate

Trotz ihrer einfachen Definition führen Pattern zu komplizierten Problemen, von denen einige durch die in der hier zusammengefassten Dissertation vorgestellten Resultate besser verstanden werden können. Auch wenn für viele dieser Probleme nur festgestellt werden konnte, dass sie beweisbar schwer sind, ist der Autor überzeugt, dass der Nutzen dieser Ergebnisse über das tiefere Verständnis von Patternsprachen hinausgeht. Zwei Beispiele, die diesen Standpunkt unterstützen, sind die in Kapitel 3 vorgestellte Erweiterung der Resultate auf ein in der Praxis weitverbreitetes Modell, sowie die in Abschnitt 2 erwähnte Anwendung in der Datenbanktheorie aus [BHLW10].

Zwei weitere Entwicklungen, die sich nach Einreichen der Dissertation ergeben haben, sind in diesem Zusammenhang erwähnenswert: Erstens wurde der in der Einleitung erwähnte

(und in Kapitel 4 angewandte) Ansatz, Resultate zu Patternsprachen in mächtigere Modelle zu übertragen und zu erweitern, inzwischen in [FS11] erfolgreich angewendet. Hierdurch konnten mehrere in [BHLW10] offen gelassene Fragen beantwortet und darüber hinausgehende Resultate erzielt werden.

Außerdem konnte der Autor bei einem Forschungsbesuch am MPII in Saarbrücken zusammen mit Timo Kötzing das Konzept der deskriptiven Generalisierung von Patternsprachen auf Unterklassen von regulären Ausdrücken übertragen. Ein Artikel über die dabei gewonnenen effizienten Algorithmen zum Approximieren von XML DTDs ist derzeit in Arbeit.

Literatur

[Ang80] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[BF10] J. Bremer und D. D. Freydenberger. Inclusion Problems for Patterns With a Bounded Number of Variables. In *Proc. DLT 2010*, Jgg. 6224 of *LNCS*, Seiten 100–111, 2010.

[BHLW10] P. Barceló, C. A. Hurtado, L. Libkin und P. T. Wood. Expressive languages for path queries over graph-structured data. In *Proc. PODS 2010*, Seiten 3–14, 2010.

[Cas02] J. Cassaigne. Unavoidable Patterns. In M. Lothaire, Hrsg., *Algebraic Combinatorics on Words*, Kapitel 3, Seiten 111–134. Cambridge University Press, Cambridge, New York, 2002.

[FR08] D. D. Freydenberger und D. Reidenbach. Bad news on decision problems for patterns. In *Proc. DLT 2008*, LNCS 5257, Seiten 327–338, 2008.

[FR09] D. D. Freydenberger und D. Reidenbach. Existence and Nonexistence of Descriptive Patterns. In *Proc. DLT 2009*, LNCS 5583, Seiten 228–239, 2009.

[FR10a] D. D. Freydenberger und D. Reidenbach. Bad news on decision problems for patterns. *Information and Computation*, 208(1):83–96, 2010.

[FR10b] D. D. Freydenberger und D. Reidenbach. Existence and nonexistence of descriptive patterns. *Theoretical Computer Science*, 411(34-36):3274 – 3286, 2010.

[FR10c] D. D. Freydenberger und D. Reidenbach. Inferring Descriptive Generalisations of Formal Languages. In *Proc. COLT 2010*, Seiten 194–206, 2010.

[FR12] D. D. Freydenberger und D. Reidenbach. Inferring Descriptive Generalisations of Formal Languages. *Journal of Computer and System Sciences*, 2012. Angenommen.

[Fre11a] D. D. Freydenberger. Extended Regular Expressions: Succinctness and Decidability. In *Proc. STACS 2011*, LIPIcs 9, Seiten 507–518, 2011.

[Fre11b] D. D. Freydenberger. *Inclusion of Pattern Languages and Related Problems*. Dissertation, Fachbereich Informatik und Mathematik, Johann Wolfgang Goethe-Universtität, Frankfurt am Main, 2011. Logos Verlag, Berlin.

[Fre12] D. D. Freydenberger. Extended Regular Expressions: Succinctness and Decidability. *Theory of Computing Systems*, 2012. Angenommen.

[FS11] D. D. Freydenberger und N. Schweikardt. Expressiveness and Static Analysis of Extended Conjunctive Regular Path Queries. In *Proc. AMW 2011*, CEUR Workshop Proceedings 749, 2011.

[Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[JKS+94] T. Jiang, E. Kinber, A. Salomaa, K. Salomaa und S. Yu. Pattern Languages With and Without Erasing. *International Journal of Computer Mathematics*, 50:147–163, 1994.

[JSSY95] T. Jiang, A. Salomaa, K. Salomaa und S. Yu. Decision Problems for Patterns. *Journal of Computer and System Sciences*, 50:53–63, 1995.

[Lag09a] J. C. Lagarias. The 3x+1 problem: An annotated bibliography (1963–1999), Aug 2009. `http://arxiv.org/abs/math/0309224`.

[Lag09b] J. C. Lagarias. The 3x+1 Problem: An Annotated Bibliography, II (2000–2009), Aug 2009. `http://arxiv.org/abs/math/0608208`.

[Nag77] T. Nagell, Hrsg. *Selected mathematical papers of Axel Thue*. Universitetsforlaget, Oslo, 1977.

[NS08] Y. K. Ng und T. Shinohara. Developments from enquiries into the learnability of the pattern languages from positive data. *Theoretical Computer Science*, 397:150–165, 2008.

[OU97] E. Ohlebusch und E. Ukkonen. On the equivalence problem for E-pattern languages. *Theoretical Computer Science*, 186:231–248, 1997.

[Rei06] D. Reidenbach. *The Ambiguity of Morphisms in Free Monoids and its Impact on Algorithmic Properties of Pattern Languages*. Dissertation, Fachbereich Informatik, Technische Universität Kaiserslautern, 2006. Logos Verlag, Berlin.

[Rei07] D. Reidenbach. An examination of Ohlebusch and Ukkonen's Conjecture on the equivalence problem for E-pattern languages. *Journal of Automata, Languages and Combinatorics*, 12:407–426, 2007.

[Sal06] K. Salomaa. Patterns, 2006. Vorlesung, 5th PhD School in Formal Languages and Applications, URV Tarragona.

[Shi82] T. Shinohara. Polynomial Time Inference of Extended Regular Pattern Languages. In *Proc. RIMS Symposia on Software Science and Engineering, Kyoto*, LNCS 147, Seiten 115–127, 1982.

[Thu06] A. Thue. Über unendliche Zeichenreihen. *Kra. Vidensk. Selsk. Skrifter. I Mat. Nat. Kl.*, 7, 1906. Nachgedruckt in [Nag77].

Dominik D. Freydenberger Geboren am 31. Oktober 1979 in Regensburg. Studium der Informatik an der TU Kaiserslautern, Abschluss als Diplom-Informatiker im März 2006. Anschließend Teilnahme an der *5th PhD School in Formal Languages and Applications* an der URV Tarragona. Von März 2007 bis Februar 2012 wissenschaftlicher Mitarbeiter am Institut für Informatik an der Goethe-Universität in Frankfurt am Main, Promotion zum Doktor der Naturwissenschaften am 27. Juni 2011. Ab März 2012 Akademischer Rat auf Zeit, ebenfalls an der Goethe-Universität.