Goethe-Universität Frankfurt am Main

Institut für Informatik

Das Aufzählungsproblem für Anfragen erster Stufe auf Strukturen von beschränktem Grad

Bachelorarbeit

eingereicht von:
Antje Peters

Betreuerin und Erstgutachterin:
Prof. Dr. Nicole Schweikardt

September 2014

Diese Arbeit befasst sich mit dem Aufzählungsproblem auf Strukturen von beschränktem Grad. Es lässt sich zeigen, dass dieses Problem in der Komplexitätsklasse CONSTANT-DELAY$_{lin}$ liegt. Dazu werden zwei unterschiedliche Beweise vorgestellt und kommentiert.

Inhaltsverzeichnis

1 Einführung **4**

2 Definitionen **5**
2.1 Ausgewählte Grundbegriffe der Logik erster Stufe 5
2.2 Das Aufzählungsproblem 6

3 Beweis nach Durand und Grandjean **8**
3.1 Das Aufzählungsproblem auf bijektiven Strukturen 8
3.2 Nützliche Eigenschaften der Klasse Constant-Delay$_{lin}$ 10
3.2.1 Konstruktion einer bijektiven Struktur aus einer Struktur von beschränktem DG-Grad 14

4 Beweis nach Kazana und Segoufin **19**
4.1 Gaifman-Lokalität 19
4.2 Details des Beweises 23

5 Zusammenfassung und Ausblick **32**
5.1 Kurze Zusammenfassung beider Beweise 32
5.2 Vergleich beider Beweise 32
5.3 Ausblick 32

1 Einführung

Aussagen treffen, Gegebenheiten spezifizieren, Beziehungen zwischen Dingen beschreiben: Dies ist nur eine kleine Auswahl von Aufgaben, mit denen sich nicht nur ein theoretischer Informatiker andauernd konfrontiert sieht. Konkret spiegeln sich diese wider in Fragestellungen wie *Umfasst meine Datenbank die für meine Zwecke nötigen Objekte?* oder *Welche Objekte aus meiner Datenbank erfüllen meine Anforderungen?*. Diese und andere Fragen lassen sich mithilfe der Logik erster Stufe (bzw. First Order- oder FO-Logik) formalisieren. Dabei sind FO-Formeln die Fragestellungen, die auf geeigneten Objekten, sog. Strukturen, ausgewertet werden. Das klassische Beispiel ist die bereits erwähnte Datenbank, an die Datenbankanfragen oder „Queries“ gestellt werden können.

Das Problem, die möglichen Lösungen für eine gegebene Anfrage und eine gegebene Datenbank zu ermitteln, das sogenannte Aufzählungsproblem, ist für viele Anwendungen des täglichen Gebrauchs von enormem Interesse. In diesem Zusammenhang geht es vor allem um die Zeit, die aufgewendet werden muss, um das Problem zu lösen. Im Allgemeinen ist diese Laufzeit alles andere als vielversprechend: Sie lautet $n^{\mathcal{O}(k)}$ und ist somit exponentiell in der Größe der k Anfrage, was für umfangreiche Datenmengen n ein Hindernis der Bedienbarkeit darstellt. Wendet man sich allerdings vom allgemeinen Fall zu spezielleren Strukturen, um das Aufzählungsproblem zu betrachten, ist es zumindest möglich, positive Prognosen für die praktische Anwendbarkeit zu machen. Die Auswertung einer Anfrage kann nämlich als dynamischer Prozess verstanden werden: So ist es auf einer Struktur von beschränktem Grad nach wie vor teuer, *alle* Lösungstupel zu ermitteln, andererseits lässt sich die Zeit, die nach Ausgabe einer Lösung verstreicht, bevor die nächste ausgegeben wird, durch eine Größe beschreiben, die nicht von der Größe der Datenbank abhängt. Diese Größe kann von der Länge der Anfrage abhängen, die allerdings als Parameter gilt, weswegen dieser Zeitbedarf als konstant angesehen wird. Weiterhin ist der Algorithmus für die Ermittlung aller Lösungen von einer besonderen Form: Er setzt sich aus einer Vorbereitungsphase mit einem Zeitbedarf linear in der Größe der Datenbank und einer Ausgabephase, in der Lösungen mit der erwähnten konstanten Verzögerung ausgegeben werden, zusammen. Folglich spricht man bei Problemen, die durch einen solchen Algorithmus gelöst werden, von Angehörigen der Klasse Constant-Delay$_{lin}$. In dieser Arbeit wird die Aussage, dass sich alle Lösungen eines Aufzählungsproblems auf Strukturen von beschränktem Grad mithilfe eines Constant-Delay$_{lin}$-Algorithmus ermitteln lassen, formalisiert. Es werden zwei verschiedene Beweisstrategien vorgestellt, die auf unterschiedlichen theoretischen Begriffen basieren. Die Beweise werden kommentiert und mit Beispielen versehen dargestellt.

Die Arbeit ist folgendermaßen aufgebaut: In Kapitel 2 werden einige unmittelbar wichtige Konzepte der Logik erster Stufe rekapituliert. Das Aufzählungsproblem und die Klasse Constant-Delay$_{lin}$ werden als zentrale Objekte der Arbeit vorgestellt. Kapitel 3 behandelt den Beweis von Durand und Grandjean, der auf der Möglichkeit der Quantorenelimination bei einer bijektiven Formel und der Umwandlung einer Struktur von beschränktem Grad in eine bijektive Struktur basiert. In Kapitel 4 wird der Beweis von Kazana und Segoufin vorgestellt, in dessen Zusammenhang der Begriff der Gaifman-Lokalität eine Rolle spielt. Im Rahmen des Beweises wird die funktionelle Form der Konstanten in Abhängigkeit der Länge der Anfrage explizit angegeben. Zum Schluss wird in Kapitel 5 das Resümee gezogen und ein Ausblick auf eine weitere Beweismöglichkeit gegeben.

2 Definitionen

In diesem Kapitel werden zunächst einige allgemeine Begriffe eingeführt, die im Laufe der Arbeit immer wieder auftreten werden. Es stellt keine vollständige Einführung in die Logik erster Stufe dar, sondern rekapituliert einige unmittelbar wichtige Konzepte. Eine umfangreiche Einführung ist zum Beispiel in [1] zu finden. Auch die in dieser Arbeit verwendete Notation folgt [1].

2.1 Ausgewählte Grundbegriffe der Logik erster Stufe

Definition 2.1 (Signatur)
Eine **Signatur** (auch Symbolmenge genannt) ist eine nichtleere Menge σ, die Konstantensymbole $c_1, \ldots, c_r$, Funktionssymbole $f_1, \ldots, f_s$ und Relationssymbole $R_1, \ldots, R_t$ $(r, s, t \in \mathbb{N})$ enthält:

$$\sigma = \{c_1, \ldots, c_r, f_1, \ldots, f_s, R_1, \ldots, R_t\} \tag{2.1}$$

Definition 2.2 (Stelligkeit)
Sei σ eine Signatur. Jedes Funktionssymbol f aus σ und jedes Relationssymbol R aus σ besitzt eine **Stelligkeit** (engl. *arity*):

$$ar(f) \in \mathbb{N}_{>0} \qquad \text{bzw.} \qquad ar(R) \in \mathbb{N}_{>0} \tag{2.2}$$

Definition 2.3 (relationale Signatur)
Eine **relationale Signatur** ist eine Signatur, die ausschließlich Relationssymbole enthält.

Definition 2.4 (unäre Signatur)
Eine **unäre Signatur** ist eine Signatur, die ausschließlich Konstantensymbole und einstellige Relations- und Funktionssymbole enthält.

Definition 2.5 (σ-Struktur)
Eine **σ-Struktur** ist ein Tupel $\mathfrak{A} = (A, \alpha)$, das aus dem nichtleeren Universum A von $\mathfrak{A}$ und einer auf σ definierten Abbildung α besteht, die die folgenden Zuordnungen vermittelt:

- Jedem Konstantensymbol $c \in \sigma$ wird ein Element $\alpha(c) \in A$ zugeordnet.
- Jedem Funktionssymbol $f \in \sigma$ wird eine Funktion $\alpha(f) : A^{ar(f)} \to A$ zugeordnet.
- Jedem Relationssymbol $R \in \sigma$ wird eine Relation $\alpha(R) \subseteq A^{ar(R)}$ zugeordnet.

Alternativ schreibt man auch:

$$\mathfrak{A} = (A, c_1^{\mathfrak{A}}, \ldots, c_r^{\mathfrak{A}}, f_1^{\mathfrak{A}}, \ldots, f_s^{\mathfrak{A}}, R_1^{\mathfrak{A}}, \ldots, R_t^{\mathfrak{A}}) \tag{2.3}$$

Definition 2.6 (Anfrage)
Sei σ eine Signatur und sei $k \geq 1$. Eine k-stellige **Anfrage** ist eine Abbildung, die jeder σ-Struktur $\mathfrak{A}$ eine k-stellige Relation $q(\mathfrak{A}) \subseteq A^k$ zuordnet.

Bemerkung 2.7
Die im Titel der Arbeit genannten Anfragen werden im Folgenden durch Formeln der Logik erster Stufe realisiert.

Definition 2.8
Sei ϕ FO-Formel. Die **Größe** $|\phi|$ ist die Anzahl der in ihr auftretenden Variablen-, Konstanten- und Funktionssymbole und der syntaktischen Symbole $\exists, \forall, \wedge, \vee, \neg, =, (,)$, und , (Komma).

Beispiel 2.9
Für $\phi \equiv \exists x \exists y R(x, y) \wedge \neg(x = y)$ ist $|\phi| = 17$.

Definition 2.10
Sei ϕ eine FO-Formel. Die Schreibweise $\phi(x_1, \ldots, x_k)$ drückt aus, dass ϕ k freie Variablen hat. Wenn die Anzahl der freien Variablen aus dem Kontext hervorgeht, schreibt man statt $\phi(x_1, \ldots, x_k)$ auch kurz $\phi(\overline{x})$.

Definition 2.11
Sei $\phi(\overline{x})$ eine FO-Formel. Die Anzahl der freien Variablen von ϕ kann durch $|\overline{x}|$ beziffert werden.

Definition 2.12
Sei $\mathfrak{A}$ eine relationale σ-Struktur. Die **Größe** $\|\mathfrak{A}\|$ von $\mathfrak{A}$ ist:

$$\|\mathfrak{A}\| := |\sigma| + |A| + \sum_{R \in \sigma} |R^{\mathfrak{A}}| \cdot ar(R^{\mathfrak{A}}) \tag{2.4}$$

wobei $|R^{\mathfrak{A}}|$ die Anzahl der Tupel in der Relation $R^{\mathfrak{A}}$ bezeichnet.

Definition 2.13 (Quantorenrang)
Sei ϕ eine FO-Formel. Der **Quantorenrang** $qr(\phi)$ von ϕ ist die maximale Anzahl von ineinander geschachtelten Quantoren, die in ϕ vorkommen. Per Induktion über den Formelaufbau ist $qr(\phi)$ wie folgt definiert:

- $qr(\phi) = 0$, falls ϕ eine atomare Formel ist.
- $qr(\phi) = qr(\psi)$, falls $\phi = \neg\psi$.
- $qr(\phi) = max\{qr(\psi_1), qr(\psi_2)\}$, falls ϕ von der Form $(\psi_1 * \psi_2)$ (mit $* \in \{\wedge, \vee, \rightarrow, \longleftrightarrow\}$).
- $qr(\phi) = qr(\psi) + 1$, falls ϕ von der Form $\exists x\psi$ oder $\forall x\psi$ ist.

2.2 Das Aufzählungsproblem

Das zentrale Objekt dieser Arbeit ist das Aufzählungsproblem der Logik erster Stufe. Ganz allgemein ist es wie folgt definiert:

Definition 2.14 (Das Aufzählungsproblem)
Das **Aufzählungsproblem** QUERY der Logik erster Stufe auf einer Klasse von Strukturen K ist wie folgt definiert:

Das Aufzählungsproblem QUERY$(\mathfrak{A}, \phi(\overline{x}))$ für FO auf K:
Sei σ eine Signatur.
Eingabe: Eine σ-Struktur $\mathfrak{A} \in K$ und eine FO$[\sigma]$-Formel $\phi(x_1, \ldots, x_k)$.
Ziel: Berechne $\phi(\mathfrak{A})$ bezüglich des Variablentupels $(x_1, \ldots, x_k)$.

Dabei ist $\phi(\mathfrak{A}) := \{(a_1, \ldots, a_k) \in A^k : \mathfrak{A} \models \phi[a_1, \ldots, a_k]\}$.

Im Folgenden wird die Komplexitätsklasse CONSTANT-DELAY$_{lin}$ formell vorgestellt. Die Tatsache, dass ein Algorithmus zwei aufeinander folgende Lösungen eines Aufzählungsproblems mit konstanter Verzögerung ausgibt, ist eine wichtige Eigenschaft für seine praktische Beherrschbarkeit.

Definition 2.15 (Die Klasse CONSTANT-DELAY$_{lin}$)
Ein Aufzählungsproblem $Q = \text{QUERY}(\mathfrak{A}, \phi(\overline{x}))$ für $\phi(\overline{x})$ auf $\mathfrak{A}$ ist lösbar mit konstanter Ausgabeverzögerung (*constant delay*) und linearer Vorverarbeitungszeit, wenn es einen Algorithmus $\mathcal{A}$ gibt, der die folgenden Bedingungen erfüllt:

a) $\mathcal{A}$ benötigt Platz, der linear in der Größe der Struktur $\mathfrak{A}$ ist: $\mathcal{O}(\|\mathfrak{A}\|)$.

b) $\mathcal{A}$ kann in die folgenden Phasen zerlegt werden:
   1) die Vorverarbeitungsphase, die Zeit $f(|\phi|)\|\mathfrak{A}\|$ benötigt (f ist eine geeignete Funktion),
   2) die Ausgabephase, in der alle Lösungstupel von Q in durch eine Konstante (bzw. durch eine Größe, die nicht von $\|\mathfrak{A}\|$ abhängt) beschränkten Zeitabständen ausgegeben werden.

Man sagt in diesem Fall, Q gehört zur Klasse CONSTANT-DELAY$_{lin}$.

Anmerkung: Eine interessante Randbeobachtung ist, dass es bis heute unklar ist, ob die Umkehrung der Aussage auch gilt, d.h., ob zu Problemen, die sich mit der o.g. Laufzeit lösen lassen auch stets ein CONSTANT-DELAY$_{lin}$-Algorithmus existiert.

Notation 2.16
Möchte man die Funktion f, die die Abhängigkeit von der Länge der Formel ϕ beschreibt, nicht präzisieren, so schreibt man für die Laufzeit einfach $\mathcal{O}_\phi(\|\mathfrak{A}\|)$.

Notation 2.17
Die Zeitkomplexität eines Algorithmus, der der Klasse CONSTANT-DELAY$_{lin}$ entspricht, wird häufig durch $f(k) \cdot (n + m)$ beschrieben. Dabei ist f eine geeignete Funktion, n die Größe der betrachteten Struktur und m die Anzahl der Lösungen des Auswertungsproblems.

3 Beweis nach Durand und Grandjean

Der älteste der in dieser Arbeit vorgestellten Beweise stammt von Arnaud Durand und Etienne Grandjean. Die Vorstellung folgt einer Veröffentlichung [2] aus dem Jahr 2007.

Die Beweisstrategie ist die folgende: Zunächst wird gezeigt, dass das Aufzählungsproblem auf bijektiven Strukturen in Constant-Delay$_{lin}$ liegt. Dabei bedient man sich der Methode der Quantorenelimination. Anschließend wird ein Algorithmus vorgestellt, der aus einer Struktur von beschränktem Grad eine bijektive Struktur erstellt. Die Laufzeit des Algorithmus ist linear in der Größe der Struktur von beschränktem Grad. Damit lässt sich schließlich die gewünschte Aussage zeigen, dass das Aufzählungsproblem auf Strukturen von beschränktem Grad in Constant-Delay$_{lin}$ liegt.

3.1 Das Aufzählungsproblem auf bijektiven Strukturen

Für spätere Betrachtungen sollen nun zwei Spezialfälle des Aufzählungsproblems beschrieben werden. Dazu werden zunächst folgende Begriffe definiert:

Definition 3.1
Eine σ-Struktur $\mathfrak{A} = (A, c_1^{\mathfrak{A}}, \ldots, c_r^{\mathfrak{A}}, f_1^{\mathfrak{A}}, \ldots, f_s^{\mathfrak{A}}, R_1^{\mathfrak{A}}, \ldots, R_t^{\mathfrak{A}})$ heißt **bijektiv**, falls $f_i^{\mathfrak{A}}$ (mit $i = 1, \ldots, s$) einstellige bijektive Funktionen und R_j (mit $j = 1, \ldots, t$) einstellige (monadische) Relationen sind.

Definition 3.2
Ein **bijektiver Term** $\tau(x)$ ist von der Form $f_1^{\epsilon_1} \circ f_2^{\epsilon_2} \circ \cdots \circ f_l^{\epsilon_l}(x)$ (abkürzende Schreibweise: $f_1^{\epsilon_1} f_2^{\epsilon_2} \ldots f_l^{\epsilon_l}(x)$). Dabei ist $l > 0$, $\epsilon_i = \pm 1$, x eine Variable und für jedes $f_i^{\epsilon_i}$ gilt:

$$f_i^{\epsilon_i} = \begin{cases} f_i \text{ (das Funktionssymbol)} & \text{falls } \epsilon_i = 1 \\ f_i^{-1} \text{ (das inverse Funktionssymbol)} & \text{falls } \epsilon_i = -1 \end{cases} \tag{3.1}$$

Der Term $\tau^{-1}(x)$ bezeichnet dementsprechend das Inverse des Terms $\tau(x)$.
Eine **bijektive atomare Formel** hat eine der folgenden Formen, wobei $\tau(x), \tau_1(x), \tau_2(x)$ bijektive Terme sind:

- $\tau_1(x) = \tau_2(y)$ (bijektive Gleichheit)
- $\tau(x) = c$, wobei c ein Konstantensymbol ist
- $U(\tau(x))$, wobei U ein einstelliges Relationssymbol ist
- $\exists_x^k \Psi(x)$ (sog. Mächtigkeitsaussage), wobei $\Psi(x)$ eine boolesche Kombination bijektiver atomarer Formeln über der Variablen x ist und das Symbol $\exists_x^k$ aussagt: „Es existieren mindestens k Werte für x, sodass“.

Ein **bijektives Literal** ist eine bijektive atomare Formel oder deren Negation.

Definition 3.3
Eine **bijektive FO[σ]-Formel** ist eine FO-Formel, die über bijektiven atomaren Formeln einer unären Signatur σ gebildet ist.

Definition 3.4
Sei σ eine relationale Signatur und $\mathfrak{A}$ eine σ-Struktur. Der **Durand-Grandjean-Grad** (kurz: **DG-Grad)** $degree_{\mathfrak{A}}(v)$ eines Elements $v \in A$ ist die Gesamtanzahl von Tupeln aus den Relationen $R_i^{\mathfrak{A}} \in \mathfrak{A}$, in denen v vorkommt. Der DG-Grad von $\mathfrak{A}$ ist $degree(\mathfrak{A}) = \max\limits_{v \in A}\{degree_{\mathfrak{A}}(v)\}$

Beispiel 3.5
Sei $\sigma = \{R_1, R_2\}$ eine relationale Signatur, $\mathfrak{A} = (\{a, b, c\}, R_1^{\mathfrak{A}}, R_2^{\mathfrak{A}})$ eine σ-Struktur mit

- $R_1^{\mathfrak{A}} = \{(a, b, c), (a, b, b), (a, c, c)\}$
- $R_2^{\mathfrak{A}} = \{(a, b, c)\}$

Element v	$degree_{\mathfrak{A}}(v)$
a	4
b	3
c	3

$\Rightarrow$ Folglich ist $degree(\mathfrak{A}) = 4$.

Definition 3.6
Sei $d \in \mathbb{N}$. Eine σ-Struktur $\mathfrak{A}$ heißt **σ-Struktur vom DG-Grad $\leq d$**, falls $degree(\mathfrak{A}) \leq d$. Man spricht in diesem Fall auch von einer Struktur von beschränktem DG-Grad.

Anmerkung: Der DG-Grad unterscheidet sich in seiner Definition von dem weiter verbreiteten Begriff Grad, der über den Gaifman-Graphen definiert ist (siehe Kapitel 4.1, der Grad des Gaifman-Graphen wird im folgenden (Gaifman-)Grad genannt). Beide Definitionen lassen sich aber eindeutig miteinander in Verbindung bringen. Die Begriffe ,,Strukturen von beschränktem DG-Grad“ und ,,Strukturen von beschränktem (Gaifman-)Grad“ beziehen sich auf dieselbe Klasse von Strukturen. Die Begriffe können als äquivalent verstanden werden. Den Zusammenhang zwischen beiden Begriffen zeigt eine einfache Abschätzung, die in folgendem Lemma formalisiert wird:

Lemma 3.7
Sei $\sigma = \{R_1, \dots R_q\}$ eine endliche relationale Signatur. Sei $m = \max\limits_{R \in \sigma}(ar(R))$. Für jede endliche σ-Struktur $\mathfrak{A}$ vom (Gaifman-)Grad g und vom DG-Grad d gilt:

a) $d \leq q \cdot (g+1)^m$

b) $g \leq d \cdot (m-1)$

D.h. $\frac{g}{m-1} \leq d \leq q \cdot (g+1)^m$.

Beweis.

a) Sei $a \in A$ beliebig. Im Gaifman-Graphen von $\mathfrak{A}$ hat a höchstens g verschiedene Nachbarn $a_1, \dots, a_g$. Jedes Tupel der Relation $R_i^{\mathfrak{A}}$ (für $i \in \{1, \dots, q\}$), in dem a vorkommt, ist aus den Elementen $\{a, a_1, \dots, a_g\}$ gebildet. Dafür kommen höchstens $(g+1)^{ar(R_i)} \leq (g+1)^m$ Tupel infrage. Insgesamt gibt es also höchstens $\sum_{i=1}^{q}(g+1)^{ar(R_i)} \leq q \cdot (g+1)^m$ Tupel in Relationen von $\mathfrak{A}$, in denen a vorkommt. Somit ist der DG-Grad $d \leq q \cdot (g+1)^m$.

b) Sei $a \in A$ beliebig. In den Relationen von $\mathfrak{A}$ gibt es maximal d Tupel $t_1, \dots, t_d$, in denen a vorkommt. Jedes dieser Tupel hat Stelligkeit $\leq m$, d.h. es enthält maximal $m-1$ verschiedene Elemente $\neq a$. Folglich ist der (Gaifman-)Grad $g \leq d \cdot (m-1)$.

□

Nun können die bereits angekündigten Spezialfälle des Aufzählungsproblems beschrieben werden:

Das Aufzählungsproblem $\text{QUERY}_{\text{FO}_{\text{bij}}}(\mathfrak{A}, \phi(\overline{x}))$ für FO auf bijektiven Strukturen:

Eingabe: Eine unäre Signatur σ, eine bijektive σ-Struktur $\mathfrak{A}$ und eine bijektive FO[σ]-Formel $\phi(x_1, \ldots, x_k)$.

Ziel: Berechne $\phi(\mathfrak{A})$ bezüglich des Variablentupels $(x_1, \ldots, x_k)$.

und

Das Aufzählungsproblem $\text{QUERY}_{\text{FO}_{\text{deg}}}(\mathfrak{A}, \phi(\overline{x}))$ für FO auf Strukturen von beschränktem Grad:

Eingabe: Eine Zahl $d \in \mathbb{N}$, eine relationale Signatur σ, eine σ-Struktur $\mathfrak{A}$ vom DG-Grad $\leq d$ und eine FO[σ]-Formel $\phi(x_1, \ldots, x_k)$.

Ziel: Berechne $\phi(\mathfrak{A})$ bezüglich des Variablentupels $(x_1, \ldots, x_k)$.

3.2 Nützliche Eigenschaften der Klasse CONSTANT-DELAY$_{lin}$

Lemma 3.8
Ein Aufzählungsproblem Q, das in der Zeit linear in Eingabe von Q lösbar ist, liegt in CONSTANT-DELAY$_{lin}$.

Beweis. Alle Lösungstupel von Q werden ausgegeben und sortiert, dabei werden möglicherweise mehrfach auftretende Lösungen entfernt. Diese Schritte werden als Vorverarbeitungsphase angesehen und benötigen die Zeit linear in der Eingabe von Q. Anschließend wird ein Lösungstupel nach dem anderen ausgegeben. Dies geschieht offensichtlich mit konstanter Verzögerung. □

Lemma 3.9
Seien Q und P zwei disjunkte Aufzählungsprobleme, das heißt, die Mengen der Lösungstupel von Q und P sind disjunkt. Sei die Vereinigung von Q und P definiert durch:

$$(Q \cup P) = \{\overline{a} : \overline{a} \text{ ist Lösungstupel von } Q \text{ oder } \overline{a} \text{ ist Lösungstupel von } P\} \tag{3.2}$$

Dann gilt auch $(Q \cup P) \in$ CONSTANT-DELAY$_{lin}$.

Beweis. Seien $\mathcal{A}$ und $\mathcal{B}$ die Algorithmen, die die Lösungstupel für Q und P ermitteln. Dann löst der folgende Algorithmus das Problem $(Q \cup P)$:

1. Vorverarbeitungsphase von $\mathcal{A}$ und $\mathcal{B}$ parallel, laut Voraussetzung in Linearzeit.
2. Ausgabe der Lösungstupel von Q und P, beschränkt durch die maximale Verzögerung von $\mathcal{A}$ und $\mathcal{B}$.

Damit erfüllt $(Q \cup P)$ die Voraussetzungen aus Definition 2.15. □

Ein, wie sich zeigen wird, wichtiges Hilfsmittel für die folgenden Überlegungen ist die Möglichkeit, beliebige bijektive FO-Formeln in quantorenfreie Formeln umzuschreiben.

Satz 3.10 (Quantorenelimination für bijektive Formeln)
Jede bijektive FO-Formel $\phi(x_1, \ldots, x_k)$ ist äquivalent zu einer booleschen Kombination $\phi'(x_1, \ldots, x_k)$ aus atomaren bijektiven FO-Formeln.

Beweis. Beachte zunächst, dass $\forall x \phi \equiv \neg(\exists x \neg \phi)$. Aus diesem Grund müssen nur Existenzquantoren eliminiert werden. O.B.d.A liege eine FO-Formel in disjunktiver Normalform vor. Da der Existenzquantor mit der Disjunktion vertauscht, kann im Folgenden eine Formel der Form

$$\phi(\overline{x}) \equiv \exists y\, (\alpha_1 \wedge \cdots \wedge \alpha_r) \tag{3.3}$$

betrachtet werden. Dabei ist jedes α_i ein bijektives Literal, das die Variablen $\overline{x}$ und y enthalten darf. Dabei brauchen Literale, die nur $\overline{x}$ enthalten, nicht betrachtet werden, da nur die Variable y durch den Existenzquantor gebunden ist. Es kann folglich angenommen werden, dass $\phi(\overline{x})$ in der folgenden Form vorliegt:

$$\phi(\overline{x}) \equiv \exists y \left(\psi(y) \wedge y =_{\epsilon_1} \tau_1(x_{i_1}) \wedge \cdots \wedge y =_{\epsilon_k} \tau_k(x_{i_k})\right) \tag{3.4}$$

die abkürzende Schreibweise $=_{\epsilon_j}$ hat dabei die folgende Bedeutung:

$$y =_{\epsilon_j} \tau_j(x_{i_j}) = \begin{cases} y = \tau_j(x_{i_j}) & \text{falls } \epsilon_j = 1 \\ \neg y = \tau_j(x_{i_j}) & \text{falls } \epsilon_j = -1 \end{cases} \tag{3.5}$$

Um nun einen Algorithmus anzugeben, der aus $\phi(\overline{x})$ eine quantorenfreie Formel $\phi'(\overline{x})$ entstehen lässt, müssen zwei unterschiedliche Fälle betrachtet werden.

Fall 1:
Es gibt mindestens einen Index j mit $\epsilon_j = 1$. Dieser Fall ist der einfachere. In diesem Fall kann für die Gleichheitsrelation $y = \tau_j(x_{i_j})$ jedes in $\phi(\overline{x})$ auftretende y durch $\tau_j(x_{i_j})$ ersetzt werden. Die dadurch entstehende Formel $\phi'(\overline{x})$ enthält kein y mehr und kann aus diesem Grund ohne Existenzquantor geschrieben werden.

Fall 2:
Sei σ eine Signatur und $\mathfrak{A}$ eine σ-Struktur. Für jedes j ist $\epsilon_j = -1$. Dies ist der kompliziertere Fall. Die Formel $\phi(\overline{x})$ lässt sich (unter der Annahme, dass o.B.d.A. $i_j = j$ für $j = 1, \ldots, k$) dann folgendermaßen schreiben:

$$\phi(\overline{x}) \equiv \exists y \left(\psi(y) \wedge \bigwedge_{j \leq k} \neg y = \tau_j(x_j)\right) \tag{3.6}$$

Sei nun $h \leq k$ die Anzahl der unterschiedlichen Terme $\tau_j(x_{i_j})$, für die $\psi(\tau_j(x_{i_j}))$ zu wahr ausgewertet werden kann. Die Formel $\phi(\overline{x})$ ist genau dann wahr, wenn es mindestens $(h+1)$ Elemente in A gibt, sodass $\exists_y^{h+1}\psi(y)$. Dann existiert nämlich mindestens ein $y \in A$, sodass sowohl $\psi(y)$ als auch $\neg y = \tau_j(x_{i_j})$ wahr ist und somit die gesamte Formel $\phi(\overline{x})$ zu wahr auswertet. Die Formel wird nun umgeschrieben zu:

$$\phi(\overline{x}) \equiv \underbrace{\bigvee_{h=0}^{k} \bigvee_{P \subseteq \{1,\ldots,k\},\, Q \subseteq P \text{ mit } |Q|=h}}_{\text{„Betrachte ein mögliches } h.\text{“}} \left[\overbrace{\underbrace{\bigwedge_{j \in Q} \psi(\tau_j(x_j))}_{\text{„}h \text{ Terme festlegen“}} \wedge \bigwedge_{i \in P} \bigvee_{j \in Q} \tau_i(x_i) = \tau_j(x_j) \wedge \underbrace{\bigwedge_{j \in \{1,\ldots,k\} \setminus P} \neg\psi(\tau_j(x_j))}_{\text{„übrige Terme fixieren“}}}^{\text{„Es gibt exakt } h \text{ unterschiedliche } \tau_j(x_j).\text{“}} \wedge \exists_y^{h+1}\psi(y) \right] \tag{3.7}$$

$\phi(\overline{x})$ ist nun eine boolesche Kombination bijektiver atomarer Formeln (vgl Definition 3.2). □

Der zentrale Satz zum Beweis von Durand und Grandjean ist der folgende:

Satz 3.11
Das Aufzählungsproblem auf bijektiven Strukturen liegt in CONSTANT-DELAY$_{lin}$, kurz: QUERY$_{\text{FO}_{\text{bij}}} \in$ CONSTANT-DELAY$_{lin}$.

Für den Beweis ist das folgende Lemma hilfreich:

Lemma 3.12
Sei $\mathfrak{A}$ eine bijektive Struktur und die Formel Ψ eine Konjunktion bijektiver Literale. Das Aufzählungsproblem für Ψ auf $\mathfrak{A}$ liegt in CONSTANT-DELAY$_{lin}$.

Beweis. Das Lemma lässt sich per Induktion über die Anzahl k der freien Variablen von $\Psi(\overline{x})$ beweisen:

$k = 1$:
Das Aufzählungsproblem für $\Psi(x)$ auf $\mathfrak{A}$ lässt sich in Zeit $\mathcal{O}_\Psi(\|A\|)$ lösen, indem man für alle Elemente $a \in A$ testet, ob $\mathfrak{A} \models \Psi[a]$ gilt. Daraus folgt nach Lemma 3.8, dass das Aufzählungsproblem in CONSTANT-DELAY$_{lin}$ liegt.

Induktionsannahme:
Die Gültigkeit von Lemma 3.12 sei für eine FO-Formel mit k freien Variablen gezeigt.

$k \to k+1$:
Nun wird das Aufzählungsproblem für $\Psi(\overline{x}, y)$ mit $\overline{x} = (x_1, \ldots, x_k)$ auf $\mathfrak{A}$ betrachtet. Das Vorgehen ähnelt dem im Beweis von Satz 3.10. Es werden zwei Fälle unterschieden:

1. Ψ enthält mindestens ein Literal der Form $\tau_1(y) = \tau_2(x_{i_0})$ mit $1 \leq i_0 \leq k$. Dies kann umgeschrieben werden zu:

$$\tau_1(y) = \tau_2(x_{i_0}) \Leftrightarrow \tau_1^{-1}\tau_1(y) = \tau_1^{-1}\tau_2(x_{i_0}) \overset{\tau=\tau_1^{-1}\tau_2}{\Leftrightarrow} y = \tau(x_{i_0}) \tag{3.8}$$

2. Ψ ist frei von derartigen Literalen.

Fall 1:
Ψ kann umgeschrieben werden zu:

$$\Psi(\overline{x}, y) \equiv \Psi_0(\overline{x}, y) \wedge y = \tau(x_{i_0}) \tag{3.9}$$

Das bedeutet, um Ψ zu *wahr* auszuwerten, muss y mit $\tau(x_{i_0})$ belegt werden. Das Auswertungsproblem lautet nun also: QUERY$_{\mathrm{FO_{bij}}}(\mathfrak{A}, \Psi_0(\overline{x}, y)\frac{\tau(x_{i_0})}{y})$.

Per Induktionsannahme kann das Problem QUERY$_{\mathrm{FO_{bij}}}(\mathfrak{A}, \Psi_0(\overline{x}, y)\frac{\tau(x_{i_0})}{y})$ durch einen Algorithmus mit konstanter Verzögerung gelöst werden. Durch Modifikation dieses Algorithmus dahin, dass statt einem Lösungstupel $\overline{a}$ das Lösungstupel $(\overline{a}, \tau(a_{i_0}))$ ausgegeben wird, gibt es also auch für Fall 1 einen Algorithmus mit konstanter Verzögerung.

Fall 2:
Dieser Fall ist der kompliziertere. Ψ kann geschrieben werden als:

$$\Psi(\overline{x}, y) \equiv \Psi_1(\overline{x}) \wedge \Psi_2(y) \wedge \bigwedge_{1 \leq i \leq r} \neg y = \tau_i(x_{j_i}) \tag{3.10}$$

mit $1 \leq j_i \leq k$, wobei $1 \leq i \leq r$

Per Induktionsannahme gibt es einen Algorithmus $\mathcal{A}_1$, der das Aufzählungsproblem für die Teilformel $\Psi_1(\overline{x})$ auf $\mathfrak{A}$ mit konstanter Verzögerung löst. Sei nun für ein Element $b \in A$ mit Q_b das Aufzählungsproblem für $\Psi(\overline{x}, y)^{\frac{b}{y}}$ auf $\mathfrak{A}$ benannt. Laut Induktionsannahme kann auch Q_b mit konstanter Verzögerung gelöst werden. Sei $\mathcal{Q}_b$ die Menge aller Lösungstupel von Q_b. Sei weiterhin mit Q_2 das Aufzählungsproblem für $\Psi_2(y)$ auf $\mathfrak{A}$ benannt, welches in Zeit $\mathcal{O}_{\Psi_2}(\|\mathfrak{A}\|)$ gelöst werden kann (vgl. Fall $k = 1$). Sei $\mathcal{Q}_2$ die Menge aller Lösungstupel von Q_2.

Falls Q_2 höchstens r Lösungen hat, gibt es einen Algorithmus $\mathcal{A}_0$, der die Lösungsmenge $\mathcal{Q}$ von Ψ als die disjunkte Vereinigung $\bigcup_{q \in \mathcal{Q}_2} \mathcal{Q}_b \times \{q\}$ ermittelt. Nach Lemma 3.9 geschieht dies mit konstanter Verzögerung.

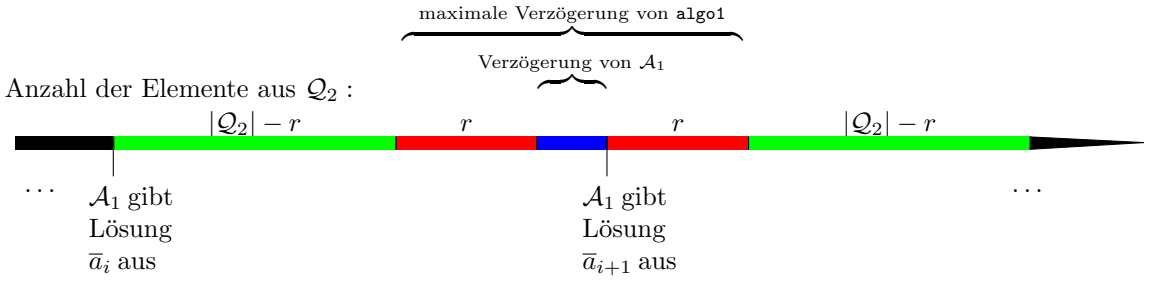
Der folgende Algorithmus formalisiert das bisher Gesagte:

Algorithmus `algo1`$(\Psi(\overline{x}, y), \mathfrak{A})$

1. Berechne die Lösungsmengen $\mathcal{Q}_2$ und die Kardinalität $|\mathcal{Q}_2|$.
2. **if** $|\mathcal{Q}_2| \leq r$
3. **then** führe Algorithmus $\mathcal{A}_0$ aus.
4. **else** führe die Vorverarbeitungsphase von $\mathcal{A}_1$ aus.
5. Für jede Lösung $\overline{a}$, die $\mathcal{A}_0$ berechnet:
6. Für jede Lösung $b \in \mathcal{Q}_2$:
7. **if** $\mathfrak{A} \not\models \left(\bigvee_{1 \leq i \leq r} y = \tau_i(x_{j_i})\right)^{\frac{b,\, a_j,\, j=1,\ldots,k}{y,\, x_{j_i}}}$ **then**
8. Gib das Lösungstupel $(\overline{a}, b)$ aus.

Algorithmus `algo1`$(\Psi(\overline{x}, y), \mathfrak{A})$ läuft offensichtlich bis einschließlich Schritt 4 in Linearzeit. Es bleibt zu zeigen, dass für den Fall, dass $|\mathcal{Q}_2| > r$, die Zeit zwischen den Ausgaben zweier Lösungstupel durch eine Konstante beschränkt ist.

Da die Anzahl der Lösungen b von Q_2, die $\mathfrak{A} \models \left(\bigvee_{1 \leq i \leq r} y = \tau_i(x_{j_i})\right)^{\frac{b,\, a_j,\, j=1,\ldots,k}{y,\, x_{j_i}}}$ erfüllen, offensichtlich durch r beschränkt ist, findet `algo1` für jede Lösung $\overline{a}$, die $\mathcal{A}_1$ berechnet, mindestens eine Gesamtlösung $(\overline{a}, b)$. Genauer gesagt gibt es für jede Lösung $\overline{a}$ sogar $|\mathcal{Q}_2| - r$ Gesamtlösungen. Folglich ist die maximale Verzögerung zwischen zwei aufeinanderfolgenden Ausgaben von Lösungen $2r$ zuzüglich der Verzögerung von $\mathcal{A}_1$. Die folgende Ablaufskizze veranschaulicht den Sachverhalt graphisch:



□

Nun lässt sich Satz 3.11 leicht beweisen.

Beweis von Satz 3.11. Betrachte das Aufzählungsproblem für $\phi(\overline{x})$ auf der bijektiven Struktur $\mathfrak{A}$. Satz 3.10 hat gezeigt, dass sich $\phi(\overline{x})$ umschreiben lässt in die disjunktive Form:

$$\phi(\overline{x}) \equiv \Psi_1(\overline{x}) \vee \cdots \vee \Psi_q(\overline{x}) \tag{3.11}$$

wobei jede Teilformel Ψ_i eine Konjunktion bijektiver Literale ist. Dank der Konstruktion der quantorenfreien Formel ϕ (vgl. die Form von (3.7)), sind die Mengen der Tupel, die die einzelnen Ψ_i erfüllen, paarweise disjunkt. Da, wie aus Lemma 3.12 hervorgeht, jedes einzelne Aufzählungsproblem für Ψ_i auf $\mathfrak{A}$ zu CONSTANT-DELAY$_{lin}$ gehört, gehört nach Lemma 3.9 $\phi(\overline{x})$ ebenfalls zu CONSTANT-DELAY$_{lin}$. □

3.2.1 Konstruktion einer bijektiven Struktur aus einer Struktur von beschränktem DG-Grad

Im Folgenden wird gezeigt, dass und wie jede Struktur von beschränktem DG-Grad als eine bijektive Struktur interpretiert werden kann. Dazu werden die folgenden beiden Strukturen betrachtet:

- Sei $d \in \mathbb{N}$. Sei ρ eine relationale Signatur. Sei $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_q^{\mathfrak{A}})$ eine ρ-Struktur vom DG-Grad $\leq d$ mit der Stelligkeit $m = \max\limits_{1 \leq i \leq q} \{ar(R_i)\}$.
- Mit $\mathfrak{A}$ wird die folgende bijektive σ-Struktur assoziiert: $\mathfrak{A}' = (A', D^{\mathfrak{A}'}, T_1^{\mathfrak{A}'}, \ldots, T_q^{\mathfrak{A}'}, g^{\mathfrak{A}'}, f_1^{\mathfrak{A}'}, \ldots, f_m^{\mathfrak{A}'})$. Die in der Signatur σ enthaltenen Symbole sind dabei die paarweise disjunkten einstelligen Relationen $D, T_1, \ldots, T_q$ und die bijektiven Funktionen $g, f_1, \ldots, f_m$.

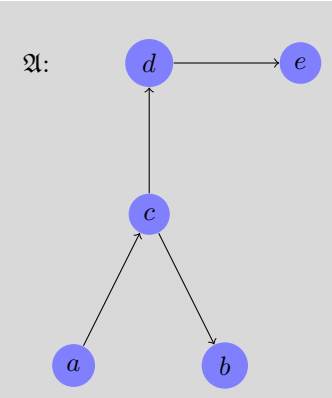
$\mathfrak{A}'$ entsteht folgendermaßen aus $\mathfrak{A}$:

- Die Relation $D^{\mathfrak{A}'}$ ist eine Kopie des Universums A.
- Jedes Element in der Relation $T_i^{\mathfrak{A}'}$ repräsentiere jeweils ein Tupel aus der Relation $R_i^{\mathfrak{A}}$. *Dieser Schritt ist in Zeit jeweils proportional zur Anzahl der Tupel der R_i durchführbar, d.h. in jedem Fall genügt $\mathcal{O}(\|\mathfrak{A}\|)$, da es sich bei $\mathfrak{A}$ um eine durch d beschränkte Struktur handelt.*
- Das Universum A' von $\mathfrak{A}'$ sei die disjunkte Vereinigung $A' = \underbrace{A \cup \left(D^{\mathfrak{A}'} \times \{1, \ldots, d\}\right)}_{=:U} \cup \underbrace{T_1^{\mathfrak{A}'} \cup \cdots \cup T_q^{\mathfrak{A}'}}_{=:T}$. *Dieser Schritt benötigt ebenfalls die Zeit $\mathcal{O}(\|\mathfrak{A}\|)$, da d eine Konstante ist.*
- Die Funktion $g^{\mathfrak{A}'}$ realisiert einen Kreis aus jeweils d Kopien jeweils eines Elements aus dem Universum von $\mathfrak{A}$. Für jedes $x \in A$ gilt somit: $g^{\mathfrak{A}'}(x) = (x, 1)$, $g^{\mathfrak{A}'}((x, i)) = (x, i + 1)$, $\ldots$, $g^{\mathfrak{A}'}((x, d - 1)) = (x, d)$ und $g^{\mathfrak{A}'}((x, d)) = x$. *Dieser Schritt benötigt die Zeit $\mathcal{O}(|A|)$, was wiederum durch $\mathcal{O}(\|\mathfrak{A}\|)$ abgedeckt wird.*
- Jede Funktion $f_i^{\mathfrak{A}'}$ verbindet die Tupelmenge T mit den Elementen aus A. Genauer gesagt: Sei $(x_1, \ldots, x_k) \in A^k$, $k \leq m$, ein Element von $R_i^{\mathfrak{A}}$, also gilt $(x_1, \ldots, x_k)$. Dann repräsentiere $t \in T_i^{\mathfrak{A}'}$ das Tupel $R_i(x_1, \ldots, x_k)$. Da $\mathfrak{A}$ eine durch den Grad d beschränkte Struktur ist, kann jedes Element aus A in höchstens d Tupeln aus $R_i^{\mathfrak{A}}$ vorkommen. Diese Vorkommen denke man sich kanonisch durchnummeriert. Sei nun $(x_1, \ldots, x_k)$ das h-te Tupel, in dem $x_j \in A, \quad j \leq k$ auftritt (klar: $h \leq d$). Setze $f_j^{\mathfrak{A}'}(t) = (x_j, h)$ und entsprechend $(f_j^{\mathfrak{A}'})^{-1}((x_j, h)) = t$. Um Bijektivität zu gewährleisten, setze $f_j^{\mathfrak{A}'}(x) = x$ für alle $x \in A' \backslash T$. *Die hierfür benötigte Zeit ist wiederum proportional zur Anzahl der Tupel in den R_i und liegt somit wieder in Zeit $\mathcal{O}(\|\mathfrak{A}\|)$.*

Somit ist $\mathfrak{A}'$ eine bijektive Struktur.

Die oben vorgestellte Vorgehensweise soll nun anhand eines übersichtlichen Beispiels verdeutlicht werden.

Beispiel 3.13
Sei die ρ-Struktur $\mathfrak{A}$ ein gerichteter Graph mit DG-Grad $d = 3$:

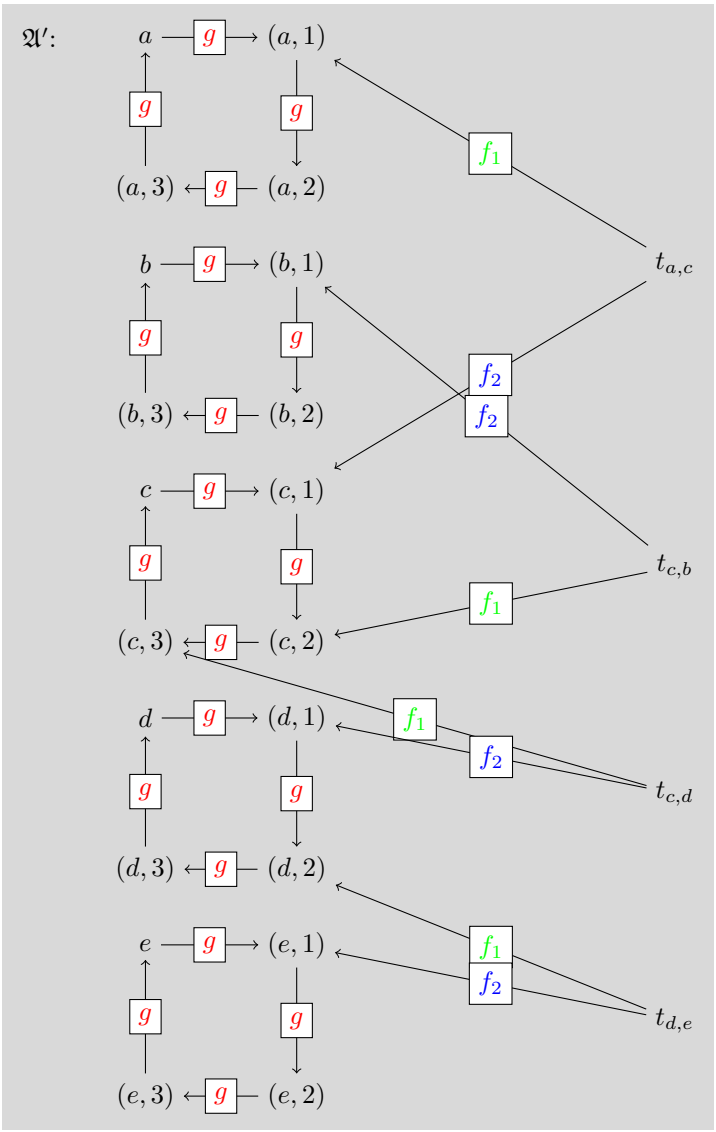


- $\rho = \{R\}$
- $\mathfrak{A} = (A, R^{\mathfrak{A}})$
- $A = \{a, b, c, d, e\}$
- $R^{\mathfrak{A}} = \{(a, c), (c, b), (c, d), (d, e)\}$, also $m = 2$

Die bijektive σ-Struktur $\mathfrak{A}'$ hat somit laut Konstruktionsvorschrift die Eigenschaften:

- $\sigma = \{D, T, g, f_1, f_2\}$
- $\mathfrak{A}' = \{A', D^{\mathfrak{A}'}, T^{\mathfrak{A}'}, g^{\mathfrak{A}'}, f_1^{\mathfrak{A}'}, f_2^{\mathfrak{A}'}\}$
- $A' = A \cup (D \times \{1, 2, 3\}) \cup T$
  $= \{a, b, c, d, e\}$
  $\cup \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3), (c, 1), (c, 2), (c, 3), (d, 1), (d, 2), (d, 3), (e, 1), (e, 2), (e, 3)\}$
  $\cup \{t_{a,c}, t_{c,b}, t_{c,d}, t_{d,e}\}$
- $D^{\mathfrak{A}'} = \{a, b, c, d, e\}$
- $T^{\mathfrak{A}'} = \{t_{a,c}, t_{c,b}, t_{c,d}, t_{d,e}\}$
- explizite Darstellung von $g^{\mathfrak{A}'}$:
  $g^{\mathfrak{A}'}(a) = (a, 1)$, $g^{\mathfrak{A}'}((a, 1)) = (a, 2)$, $g^{\mathfrak{A}'}((a, 2)) = (a, 3)$, $g^{\mathfrak{A}'}((a, 3)) = a$
  $g^{\mathfrak{A}'}(b) = (b, 1)$, $g^{\mathfrak{A}'}((b, 1)) = (b, 2)$, $g^{\mathfrak{A}'}((b, 2)) = (b, 3)$, $g^{\mathfrak{A}'}((b, 3)) = b$
  $g^{\mathfrak{A}'}(c) = (c, 1)$, $g^{\mathfrak{A}'}((c, 1)) = (c, 2)$, $g^{\mathfrak{A}'}((c, 2)) = (c, 3)$, $g^{\mathfrak{A}'}((c, 3)) = c$
  $g^{\mathfrak{A}'}(d) = (d, 1)$, $g^{\mathfrak{A}'}((d, 1)) = (d, 2)$, $g^{\mathfrak{A}'}((d, 2)) = (d, 3)$, $g^{\mathfrak{A}'}((d, 3)) = d$
  $g^{\mathfrak{A}'}(e) = (e, 1)$, $g^{\mathfrak{A}'}((e, 1)) = (e, 2)$, $g^{\mathfrak{A}'}((e, 2)) = (e, 3)$, $g^{\mathfrak{A}'}((e, 3)) = e$
- explizite Darstellung von $f_1^{\mathfrak{A}'}$ und $f_2^{\mathfrak{A}'}$: betrachte alle Tupel in $R^{\mathfrak{A}}$ und nummeriere sie durch: z.B. (a, c): 1. Auftreten von Element a, 1. Auftreten von Element c, (c, b): 2. Auftreten von Element c, 1. Auftreten von Element b u.s.w.
  Somit: $f_1^{\mathfrak{A}'}(t_{a,c}) = (a, 1)$ $f_2^{\mathfrak{A}'}(t_{a,c}) = (c, 1)$
  $f_1^{\mathfrak{A}'}(t_{c,b}) = (c, 2)$ $f_2^{\mathfrak{A}'}(t_{c,b}) = (b, 1)$
  $f_1^{\mathfrak{A}'}(t_{c,d}) = (c, 3)$ $f_2^{\mathfrak{A}'}(t_{c,d}) = (d, 1)$
  $f_1^{\mathfrak{A}'}(t_{d,e}) = (d, 2)$ $f_2^{\mathfrak{A}'}(t_{d,e}) = (e, 1)$

Die bijektiven Struktur lässt sich folgendermaßen skizzieren (Anmerkung: die Eigenkanten der Funktionen $f_1^{\mathfrak{A}'}$ und $f_2^{\mathfrak{A}'}$ sind der Übersichtlichkeit halber nicht gezeigt):



Wie im Zuge der Konstruktion bereits angemerkt, erfolgt die Konstruktion einer bijektiven Struktur aus einer Struktur von beschränktem Grad in Linearzeit. Dies wird in einem eigenen Lemma präzisiert:

Lemma 3.14
Die Konstruktion einer bijektiven Struktur $\mathfrak{A}'$ aus einer Struktur $\mathfrak{A}$ vom DG-Grad $\leq d$ erfolgt in Zeit $\mathcal{O}_{\rho,d}(\|\mathfrak{A}\|)$.

Beweis. Betrachte die Struktur $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_q^{\mathfrak{A}})$ und $\mathfrak{A}' = (A', D^{\mathfrak{A}'}, T_1^{\mathfrak{A}'}, \ldots, T_q^{\mathfrak{A}'}, g^{\mathfrak{A}'}, f_1^{\mathfrak{A}'}, \ldots, f_m^{\mathfrak{A}'})$ wie zu Beginn von Abschnitt 3.2.1 beschrieben. Es muss lediglich die Größe der beiden Strukturen $\mathfrak{A}$ und $\mathfrak{A}'$ verglichen werden, da die einzelnen Schritte der Konstruktion jeweils in $\mathcal{O}(\|\mathfrak{A}\|)$ liegen. Die Notation $\Theta_k(n)$ bedeutet dabei $\Theta(f(k) \cdot n)$ für eine beliebige Funktion f.

Die Größe von $\mathfrak{A}$ ist

$$\|\mathfrak{A}\| = |A| + q + \sum_{i=1}^{q} ar(R_i^{\mathfrak{A}})|R_i^{\mathfrak{A}}| = \Theta_\rho(|A| + \sum_{i=1}^{q} |R_i^{\mathfrak{A}}|) \tag{3.12}$$

Weiterhin gilt für A' per Konstruktion (beachte: $|T_i| = |R_i|$ für alle i):

$$\begin{aligned} |A'| &= |A \cup (D \times \{1, \ldots, d\}) \cup T_1 \cup \cdots \cup T_k| \\ &= |A| + d \cdot |A| + \sum_{i=1}^{q} |R_i^{\mathfrak{A}}| \\ &= (d+1) \cdot |A| + \sum_{i=1}^{q} |R_i^{\mathfrak{A}}| \\ &= \Theta_{\rho,d}(\|\mathfrak{A}\|) \end{aligned} \tag{3.13}$$

Da $g, f_1, \ldots, f_m$ einstellige Funktionen sind, die von A' nach A' abbilden, gilt für sie: $|g| = |f_1| = \cdots = |f_m| = |A'|$. Folglich findet man für die Größe von $\mathfrak{A}'$ insgesamt:

$$\begin{aligned} \|\mathfrak{A}'\| &= (d+1) \cdot (m+2) \cdot |A| + (d+2) \cdot \sum_{i=1}^{q} |R_i^{\mathfrak{A}}| \\ &= \Theta_{\rho,d}(\|\mathfrak{A}\|) \end{aligned} \tag{3.14}$$

□

Die Äquivalenz der Struktur von beschränktem Grad $\mathfrak{A}$ und der aus ihr konstruierten bijektiven Struktur $\mathfrak{A}'$ wird im Folgenden bewiesen.

Lemma 3.15
Sei ρ eine relationale Signatur Sei $\mathfrak{A}$ eine FO[ρ]-Struktur vom DG-Grad $\leq d$. Sei $\mathfrak{A}'$ die mit $\mathfrak{A}$ assoziierte bijektive σ-Struktur, die anhand der oben vorgestellten Vorschrift konstruiert wurde. Die Auswertung einer FO-Formel auf $\mathfrak{A}$ lässt sich auf $\mathfrak{A}'$ simulieren.

Beweis. Der Beweis erfolgt in Form einer Reduktion. Es wird die Auswertung einer Formel $\phi(\overline{x})$ auf der Struktur $\mathfrak{A}$ vom DG-Grad $\leq d$ durch die Auswertung einer Formel $\psi(\overline{x})$ auf der bijektiven Struktur $\mathfrak{A}'$ simuliert.

Zunächst wird dafür $\mathfrak{A}'$ wie oben angegeben aus $\mathfrak{A}$ konstruiert.

Jeder FO[ρ]-Formel $\phi(x_1, \ldots, x_p)$ werde die FO[σ]-Formel $\phi'(x_1, \ldots, x_p)$ zugeordnet, wobei folgende Ersetzungen vorgenommen werden:

- $\forall v \to \forall v D(v)$ und $\exists v \to \exists v D(v)$
- $R_i(x_1, \ldots, x_k) \to \theta_i(x_1, \ldots, x_k)$

wobei die Formel θ_i jedem Relationssymbol $R_i \in \rho$ mit Stelligkeit k zugeordnet wird:

$$\theta_i(x_1, \ldots, x_k) \equiv \exists t \left(T_i(t) \bigwedge_{1 \leq j \leq k} \bigvee_{1 \leq h \leq d} f_j(t) = g^h(x_j) \right) \tag{3.15}$$

Damit gilt also für alle k-Tupel $(a_1, \ldots, a_k) \in D^k$:

$$\mathfrak{A} \models R_i[a_1, \ldots, a_k] \Leftrightarrow \mathfrak{A}' \models \theta_i[a_1, \ldots, a_k] \tag{3.16}$$

Somit ergibt sich für alle p-Tupel $(a_1, \ldots, a_p) \in D^p$:

$$\mathfrak{A} \models \phi[a_1, \ldots, a_p] \Leftrightarrow \mathfrak{A}' \models \phi'[a_1, \ldots, a_p] \tag{3.17}$$

Nun muss noch gewährleistet werden, dass eine Formel auf $\mathfrak{A}'$ nur zu wahr ausgewertet wird, wenn die Elemente, über die in einer (p-Stelligen) Relation „gesprochen wird“, auch tatsächlich in $\mathfrak{A}$ existieren. Mit anderen Worten, man will lediglich die Tupelmenge $\phi'(\mathfrak{A}') \cap A^p$ erlauben.

Somit gilt

$$\text{Sei } \psi(x_1, \ldots, x_p) \equiv \phi'(x_1, \ldots, x_p) \wedge \bigwedge_{i \leq p} D(x_i) \text{ , so} \tag{3.18}$$

$$\phi(\mathfrak{A}) = \psi(\mathfrak{A}') \tag{3.19}$$

□

Nun kann endlich die finale Aussage getroffen werden:

Satz 3.16
$\text{QUERY}_{\text{FO}_{\text{deg}}}$ gehört zu $\text{CONSTANT-DELAY}_{lin}$.

Beweis. Betrachte das Aufzählungsproblem $\text{QUERY}_{\text{FO}_{\text{deg}}}(\mathfrak{A}, \phi(\overline{x}))$ für $\phi(\overline{x})$ auf der durch d beschränkten Struktur $\mathfrak{A}$. Sei der $\mathcal{A}$ der $\text{CONSTANT-DELAY}_{lin}$-Algorithmus, der das Aufzählungsproblem $\text{QUERY}_{\text{FO}_{\text{bij}}}(\mathfrak{A}', \psi(\overline{x}))$ löst.

- Schritt 1: Berechne $\psi(\overline{x})$ wie im Beweis von Lemma 3.15 gezeigt.
- Schritt 2: Konstruiere die bijektive Struktur $\mathfrak{A}'$ aus $\mathfrak{A}$.
- Schritt 3: Löse das Aufzählungsproblem für $\psi(\overline{x})$ auf $\mathfrak{A}'$ mit Hilfe von $\mathcal{A}$.

Schritt 1 ist in $\mathcal{O}_{\phi,d}(1)$ durchführbar. Die Stelligkeit der Relationen in ρ ist im Index ρ von $\mathcal{O}$ in Lemma 3.14 kodiert. Aufgrund der Überlegungen, die zu Gleichungen (3.18) und (3.19) führen, wird der mit der Stelligkeit verbundene Zeitaufwand aber durch den Ausdruck $\mathcal{O}_{\phi}(\|\mathfrak{A}\|)$ abgedeckt. Folglich weiß man mit Lemma 3.14, dass Schritt 2 in $\mathcal{O}_{\phi,d}(\|\mathfrak{A}\|)$ passiert. Diese beiden Zeiten ergeben zusammen mit der Vorverarbeitungsphase von $\mathcal{A}$, welche Zeit linear in der Größe von $\mathfrak{A}$ benötigt, den Zeitbedarf der Vorverarbeitungsphase $\mathcal{O}_{\phi,d}(\|\mathfrak{A}\|)$. Algorithmus $\mathcal{A}$ gibt die Lösungstupel dann mit konstanter Verzögerung aus. □

Damit ist der erste der drei Beweise abgeschlossen. Der Beweis hat eine andere Gruppe dazu angeregt, eine alternative Beweisstrategie zu entwickeln.

4 Beweis nach Kazana und Segoufin

Als Reaktion auf den im vorherigen Abschnitt vorgestellten Beweis präsentierten Wojciech Kazana und Luc Segoufin im Jahr 2011 eine alternative Beweisstrategie [3]. Der Beweis bedient sich der Idee der Gaifman-Lokalität für FO-Formeln. Anders als Durand und Grandjean geben Kazana und Segoufin explizit die Laufzeit der konstanten Verzögerung als dreifach exponentiell in der Größe der Eingabeformel an.

Der Beweis wird direkt für Strukturen von beschränktem Grad ohne Umweg über die Umwandlung in bijektive Strukturen geführt.

4.1 Gaifman-Lokalität

Die Möglichkeit, Strukturen mit relationaler Signatur einen Gaifman-Graphen zuzuordnen, erleichtert es, Aussagen über die Struktur zu treffen. Zudem ermöglicht die Einführung eines Distanzmaßes unter anderem die einfache Veranschaulichung eines Sachverhalts.

Definition 4.1 (Gaifman-Graph)
Sei σ eine relationale Signatur. Sei $\mathfrak{A}$ eine σ-Struktur.

a) Der **Gaifman-Graph** $G(\mathfrak{A})$ von $\mathfrak{A}$ ist der ungerichtete Graph mit der Knotenmenge $V = A$ und der Kantenmenge

$$E = \big\{\{u,v\} : u \neq v \text{ und es gibt ein } R \in \sigma \text{ und ein Tupel } (a_1,\ldots,a_k) \in R^{\mathfrak{A}} \text{ mit } u,v \in \{a_1,\ldots,a_k\}\big\} \tag{4.1}$$

b) Der **(Gaifman-)Grad** ist der Grad des Gaifman-Graphen, also die Zahl $\max\limits_{a\in A} \big|\{b \in A : \{a,b\} \text{ ist Kante von } G(\mathfrak{A})\}\big|$.

c) Die **Distanz** $\delta(a,b)$ zwischen zwei gegebenen Elementen $a,b \in A$ ist die Länge des kürzesten Weges zwischen a und b in $G(\mathfrak{A})$. Falls es zwischen a und b in $G(\mathfrak{A})$ keinen Weg gibt, so ist $\delta(a,b) = \infty$.

d) Die **Distanz zwischen zwei Tupeln** $\overline{a} = (a_1,\ldots,a_k) \in \mathfrak{A}$ und $\overline{b} = (b_1,\ldots,b_l) \in \mathfrak{A}$ ist

$$\delta(\overline{a},\overline{b}) \equiv min\,\{\delta(a_i,b_j) : 1 \leq i \leq k,\ 1 \leq j \leq l\} \tag{4.2}$$

e) Für gegebenes $r \in \mathbb{N}$ und ein Tupel $\overline{a} \in A^k$ sei $N_r^{\mathfrak{A}}(\overline{a})$ die Menge von Elementen aus A, deren Distanz zu $\overline{a}$ kleiner oder gleich r ist. Die durch $N_r^{\mathfrak{A}}(\overline{a})$ induzierte Substruktur von $\mathfrak{A}$ ist die **r-Umgebung** $\mathcal{N}_r^{\mathfrak{A}}(\overline{a})$ von $\overline{a}$. Im Folgenden wird häufig von der (rk)-Umgebung eines Elements die Rede sein, was vor allem bei der Beschreibung der Umgebung der k freien Variablen einer Formel Sinn ergibt.

f) Eine Formel $\phi(x_1,\ldots,x_k)$ heißt **gebunden um** x_1, wenn $\phi(x_1,\ldots,x_k)$ logisch impliziert, dass $x_2,\ldots,x_k$ in der (rk)-Umgebung von x_1 liegen, wobei $r = 7^{qr(\phi)}$.

An dieser Stelle wird die in Kapitel 2.2 angekündigte Definition einer durch den Grad des Gaifman-Graphen beschränkten Struktur formuliert.

Definition 4.2 (Beschränkte Struktur)
Sei $d \in \mathbb{N}$. Sei $\mathfrak{A}$ eine Struktur. $\mathfrak{A}$ heißt **beschränkt durch** d, wenn ihr Gaifman-Graph $G(\mathfrak{A})$ einen durch d beschränkten Grad hat.

Im Folgenden wird der Satz von Gaifman angegeben und angewendet. In seinem Kontext spielen der Begriff der Gaifman-Normalform und der Lokalität eine wichtige Rolle. Zunächst werden diese Begriffe näher betrachtet.

Definition 4.3 (lokale Formel)
Sei σ eine relationale Signatur. Sei $\phi(\overline{x}) = \phi(x_1, \ldots, x_k)$ eine FO[σ]-Formel. $\phi(\overline{x})$ heißt **r-lokal um $\overline{x}$**, falls für alle σ-Strukturen $\mathfrak{A}$ und alle Tupel $\overline{a} \in A^k$ ($k \in \mathbb{N}$) gilt:

$$\mathfrak{A} \models \phi[\overline{a}] \iff \mathcal{N}_r^{\mathfrak{A}}(\overline{a}) \models \phi[\overline{a}] \tag{4.3}$$

Eine FO[σ]-Formel $\phi(\overline{x})$ heißt **lokal um** $\overline{x}$, falls es eine Zahl $r \in \mathbb{N}$ gibt, so dass $\phi(\overline{x})$ r-lokal um $\overline{x}$ ist.

Definition 4.4 (basislokaler Satz)
Sei σ eine relationale Signatur und seien $l, r \in \mathbb{N}$ mit $l \geq 1$. Ein FO[σ]-Satz heißt **basislokal** mit den Parametern l, r, wenn er von der Form

$$\exists x_1 \ldots \exists x_l \left(\left(\bigwedge_{1 \leq i < j \leq l} dist(x_i, x_j) > 2r \right) \wedge \left(\bigwedge_{i=1}^{l} \psi(x_i) \right) \right) \tag{4.4}$$

ist, wobei $\psi(x)$ eine FO[σ]-Formel ist, die r-lokal um x ist und $dist(x_i, x_j)$ die Distanz $\delta(x_i, x_j)$ im Gaifman-Graphen angibt.

Definition 4.5 (Gaifman-Normalform)
Sei σ eine relationale Signatur. Ein FO[σ]-Satz ϕ ist in **Gaifman-Normalform (GNF)**, falls ϕ eine boolesche Kombination basislokaler Sätze ist. Eine FO[σ]-Formel $\psi(\overline{x})$ ist in GNF, falls $\psi(\overline{x})$ eine boolesche Kombination von basislokalen Sätzen und von Formeln, die lokal um $\overline{x}$ sind, ist.

Die Äquivalenz einer FO-Formel zu einer Formel in GNF wird im Satz von Gaifman formalisiert. In dieser Arbeit wird der Satz ohne Beweis angegeben.

Satz 4.6 (Satz von Gaifman (siehe z.B. [4]))
Sei σ eine relationale Signatur. Jede FO[σ]-Formel $\phi(\overline{x})$ ist äquivalent zu einer FO[σ]-Formel in Gaifman-Normalform. Weiterhin gilt:

- Es gibt einen Algorithmus, der $\phi(\overline{x})$ in GNF transformiert.
- Falls ϕ ein FO-Satz ist, so ist seine GNF eine boolesche Kombination basislokaler Sätze.
- Falls gilt: $qr(\phi) = k$ und $|\overline{x}| = n$, so erfüllen für die Parameter l und r der GNF: $r \leq 7^k$, $l \leq k + n$.

Der für das Folgende wichtigste Aspekt des Beweises ist der für die GNF angenommene Parameter r, der als Funktion der Länge der Formel $\phi(\overline{x})$ angegeben werden kann.

Satz 4.7 (Gaifman-Lokalität für FO)
Sei σ eine relationale Signatur, sei $\mathfrak{A}$ eine σ-Struktur und sei $\phi(x_1, \ldots, x_k)$ eine FO-Formel. Für alle Tupel $\overline{a}, \overline{b} \in A^k$ gilt:

$$\text{Falls } \mathcal{N}_r^{\mathfrak{A}}(\overline{a}) \simeq \mathcal{N}_r^{\mathfrak{A}}(\overline{b}), \text{ so} \qquad \mathfrak{A} \models \phi[\overline{a}] \iff \mathfrak{A} \models \phi[\overline{b}], \text{ wobei } r = 7^{qr(\phi)} \tag{4.5}$$

Der Beweis basiert im Wesentlichen auf dem Satz von Gaifman.

Beweis. Gemäß dem Satz von Gaifman ist $\phi(\overline{x})$ äquivalent zu einer Formel $\phi'(\overline{x}) = \phi(x_1, \ldots, x_k)$ in Gaifman-Normalform. $\phi'(\overline{x})$ ist eine boolesche Kombination von basislokalen Sätzen $\chi_1, \ldots, \chi_s$ und lokalen Formeln $\psi_1(\overline{x}), \ldots, \psi_t(\overline{x})$ ($s, t \in \mathbb{N}$). Sei $r = 7^{qr(\phi)}$, sodass jede der lokalen Formeln $\psi_i(\overline{x})$ in der Gaifman-Normalform r-lokal um $\overline{x}$ ist. Nun sei $\mathfrak{A}$ eine beliebige Struktur und $\overline{a}, \overline{b} \in A^k$ seien beliebige Tupel, so dass

$$\mathcal{N}_r^{\mathfrak{A}}(\overline{a}) \simeq \mathcal{N}_r^{\mathfrak{A}}(\overline{b}) \tag{4.6}$$

Zu zeigen ist nun, dass

$$\mathfrak{A} \models \phi'[\overline{a}] \Leftrightarrow \mathfrak{A} \models \phi'[\overline{b}] \tag{4.7}$$

Wegen der GNF von ϕ' genügt es zu zeigen, dass

1. für alle $i \in \{1, \ldots, s\}$ gilt $\mathfrak{A} \models \chi_i[\overline{a}] \Leftrightarrow \mathfrak{A} \models \chi_i[\overline{b}]$
2. für alle $i \in \{1, \ldots, t\}$ gilt $\mathfrak{A} \models \psi_i[\overline{a}] \Leftrightarrow \mathfrak{A} \models \psi_i[\overline{b}]$

(1) gilt ganz offensichtlich, da χ_i als Satz überhaupt nicht von den Tupel $\overline{a}$ und $\overline{b}$ abhängt. (2) gilt wegen (4.6) und da ψ_i r-lokal um $\overline{x}$ ist. □

Satz 4.8 ([5])
Sei $d \in \mathbb{N}$ beliebig, aber fest. Sei ϕ ein FO-Satz und sei $\mathfrak{A}$ eine Struktur, deren Gaifman-Graph $G(\mathfrak{A})$ einen durch d beschränkten Grad hat. Ob gilt $\mathfrak{A} \models \phi$, ist in Zeit $2^{2^{2^{\mathcal{O}(|\phi|)}}} \cdot \|\mathfrak{A}\|$ entscheidbar.

Beweis. Der Beweis folgt dem Vorgehen in [5]. Der folgende Algorithmus löst das Entscheidungsproblem für ϕ auf $\mathfrak{A}$. In jedem Durchlauf des Algorithmus wird Satz 4.7 benutzt.

Algorithmus `algo2`(ϕ, $\mathfrak{A}$)

1. **if** ϕ enthält keine Quantoren, **then**
2. **if** $\mathfrak{A} \models \phi$ **then** akzeptiere, **else** verwerfe
   // *Im Folgenden sei* $\phi \equiv Qx\psi(x)$, $Q \in \{\forall, \exists\}$
3. Berechne $r = 7^{qr(\phi)}$.
4. Berechne die Menge $X \subseteq A$ von Repräsentanten von Isomporphietypen von r-Umgebungen von $\mathfrak{A}$.
5. Berechne rekursiv `algo2`($\psi(a)$, $\mathfrak{A}$) für alle $a \in X$:
6. **if** $\phi \equiv \exists x\psi(x)$ **then**
7. **if** $\mathfrak{A} \models \psi(a)$ für mindestens ein a **then** akzeptiere, **else** verwerfe
8. **if** $\phi \equiv \forall x\psi(x)$ **then**
9. **if** $\mathfrak{A} \models \psi(a)$ für alle a **then** akzeptiere, **else** verwerfe

Sei $R(n, p, q)$ die Laufzeit von `algo2`(ϕ, $\mathfrak{A}$), dabei ist $n = |\mathfrak{A}|$, $q = |\phi|$ und p die Länge des quantorenfreien Teils von ϕ. Sei $r = 7^{qr(\phi)}$ und sei

$$s(q) = \max_{a \in A} \|\mathcal{N}_r^{\mathfrak{A}}(a)\| \tag{4.8}$$

die maximale Größe einer r-Umgebung. Sei $t(q)$ die Anzahl unterschiedlicher r-Umgebungstypen. Die obere Grenze von s und t hängt lediglich vom Grad des Gaifman-Graphen von $\mathfrak{A}$ ab, der als konstant vorausgesetzt wird.

Jetzt wird der Algorithmus `algo2` genauer betrachtet. Zeile 1 benötigt konstante Zeit. Falls die Überprüfung in Zeile 2 durchgeführt wird, benötigt sie die Zeit $\mathcal{O}(p \cdot n)$. Ansonsten wird zu Zeile 3 gesprungen und diese wird in konstanter Zeit abgearbeitet. In Zeile 4 wird die Liste aller Elemente $a \in A$ erstellt, die jeweils eine Äquivalenzklasse von Isomorphietypen von r-Umgebungen $N_r^{\mathfrak{A}}(a)$ repräsentieren. Die Größe der Liste kann niemals $t(q)$ übersteigen, denn für jedes a wird jeweils $N_r^{\mathfrak{A}}(a)$ berechnet und dann ermittelt, ob sie sich bereits in der Liste befindet. Falls nicht, wird sie hinzugefügt. Es wird die Zeit $\mathcal{O}(n \cdot f(s(q)) \cdot t(q))$ benötigt, wenn man annimmt, dass die Zeit,

die benötigt wird, um die Isomorphie von Strukturen der Größe m zu überprüfen, mit $f(m)$ zu Buche schlägt. Die Schleife, die in Zeile 5 beginnt, benötigt die Zeit $\mathcal{O}(t(q)) + t(q) \cdot R(n, p, q-1)$. Insgesamt ergibt sich damit für die Laufzeit R und geeignete Konstanten $c_1, c_1 \in \mathbb{R}$ die folgende Rekursionsvorschrift:

$$R(n, p, 0) \leq c_1 \cdot p \cdot n \tag{4.9}$$
$$R(n, p, q) \leq c_1 \cdot n \cdot f(s(q)) \cdot t(q) + t(q) \cdot R(n, p, q-1) \quad \text{für } q \geq 1 \tag{4.10}$$

Um diese Gleichung zu lösen bedient man sich dem folgenden Lemma:

Lemma 4.9
Seien $F, g, h : \mathbb{N} \to \mathbb{N}$ mit

$$\begin{aligned} F(0) &\leq g(0) \\ F(m) &\leq g(m) + h(m) \cdot F(m-1) \end{aligned} \tag{4.11}$$

für alle $m \in \mathbb{N}$. Dann gilt für alle $m \in \mathbb{N}$:

$$F(m) \leq \sum_{i=0}^{m} g(i) \cdot \prod_{j=i+1}^{m} h(j) \tag{4.12}$$

Beweis. Per Induktion lässt sich das Lemma leicht beweisen.
Induktionsanfang $m = 0$:

$$F(0) \leq \sum_{i=0}^{0} g(i) \cdot \underbrace{\prod_{j=1}^{0} h(j)}_{\equiv 1 \text{ (leeres Produkt)}} = g(0) \tag{4.13}$$

Induktionsschritt $m \to m+1$:

$$F(m+1) \overset{(4.11)}{\leq} g(m+1) + h(m+1) \cdot F(m) \tag{4.14}$$
$$\overset{4.12}{\leq} g(m+1) + h(m+1) \cdot \sum_{i=0}^{m} g(i) \cdot \prod_{j=i+1}^{m+1} h(j) \tag{4.15}$$
$$\equiv g(m+1) \underbrace{\prod_{j=m+2}^{m+1} h(j)}_{\equiv 1} \cdot + \sum_{i=0}^{m} g(i) \cdot \prod_{j=i+1}^{m+1} h(j) \tag{4.16}$$
$$= \sum_{i=0}^{m+1} g(i) \cdot \prod_{j=i+1}^{m+1} h(j) \tag{4.17}$$

□

Wendet man dieses Lemma auf R an, so findet man mit $g(0) = c_1 \cdot p \cdot n$, $g(q) = c_2 \cdot n \cdot f(s(q)) \cdot t(q)$ und $h(q) = t(q)$:

$$R(n,p,q) \leq g(0) \cdot \prod_{j=1}^{q} h(j) + \sum_{i=1}^{q} g(i) \cdot \prod_{j=i+1}^{q} h(j) \tag{4.18}$$

$$= c_1 \cdot p \cdot n \cdot \prod_{j=1}^{q} t(j) + \sum_{i=1}^{q} c_2 \cdot n \cdot f(s(i)) \cdot t(i) \cdot \prod_{j=i+1}^{q} t(j) \tag{4.19}$$

$$\overset{\text{wg. } t(1)\geq 1}{\leq} c_1 \cdot p \cdot n \cdot \prod_{j=1}^{q} t(j) + \sum_{i=1}^{q} c_2 \cdot n \cdot f(s(i)) \cdot t(i) \cdot t(1) \cdot \prod_{j=i+1}^{q} t(j) \tag{4.20}$$

$$= \prod_{j=1}^{q} t(j) \left(c_1 \cdot p \cdot n + \sum_{i=1}^{q} c_1 \cdot n \cdot f(s(i)) \right) \tag{4.21}$$

Die Größe einer r-Umgebung ist durch d^r beschränkt, d.h.

$$s(q) \leq d^r = d^{2^q} = 2^{2^{\mathcal{O}(q)}} \tag{4.22}$$

Die obere Grenze der Anzahl der unterschiedlichen Isomorphietypen von r-Umgebungen $t(q)$ ist

$$t(q) \leq d^{\mathcal{O}(q \cdot r)} \tag{4.23}$$

dies ergibt sich aus der folgenden Überlegung: Die Anzahl von unterschiedlichen r-Umgebungstypen eines Gaifman-Graphen mit m Knoten und d_1 einstelligen, d_2 zweistelligen,... und d_d d-stelligen Relationssymbolen ist durch $d^{d_1 \cdot m} \cdot \ldots \cdot d^{d_d \cdot m} \leq d^{d \cdot q \cdot r}$ beschränkt, denn pro Relationssymbol müssen höchstens q Symbole berücksichtigt werden, da ϕ weniger als q freie Variablen besitzt. Weiter findet man

$$\prod_{j=1}^{q} t(j) \leq \prod_{j=1}^{q} d^{\mathcal{O}(q \cdot r)} \leq d^{\mathcal{O}(q \cdot \sum_{j=1}^{q} d^j)} \leq 2^{2^{\mathcal{O}(q)}} \tag{4.24}$$

Für den zweiten Teil von (4.18) ergibt sich:

$$\left(c_1 \cdot p \cdot n + \sum_{i=1}^{q} c_1 \cdot n \cdot f(s(i)) \right) \leq \mathcal{O}(2^{s(q)} \cdot n) \tag{4.25}$$

und damit für die Laufzeit von R insgesamt:

$$R(n,p,q) \leq 2^{2^{2^{\mathcal{O}(q)}}} \cdot n = 2^{2^{2^{\mathcal{O}(|\phi|)}}} \cdot \|\mathfrak{A}\| \tag{4.26}$$

□

4.2 Details des Beweises

Die folgenden Begriffe finden spezielle Verwendung im Beweis und werden deshalb etwas ausführlicher dargestellt.

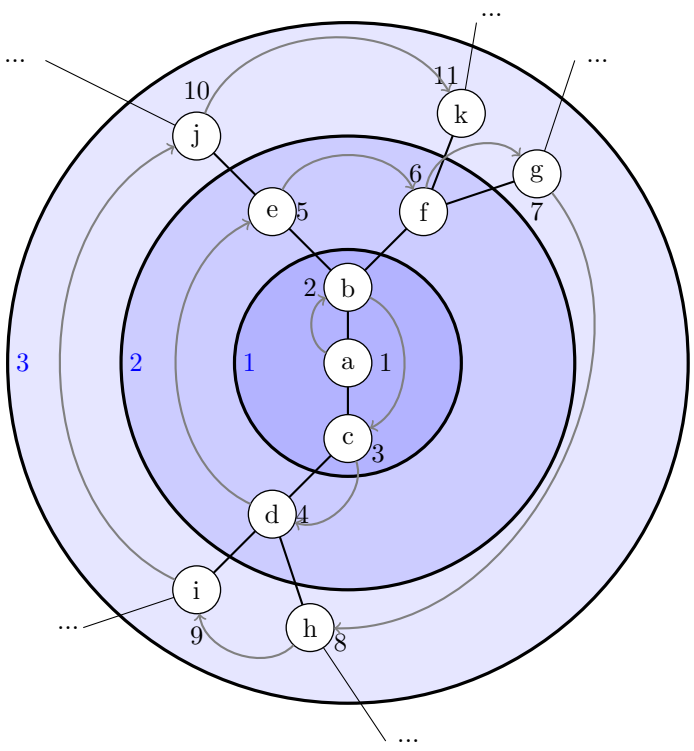
Definition 4.10
Sei $d \in \mathbb{N}$. Sei σ eine relationale Signatur und sei $\mathfrak{A}$ im Folgenden eine durch d beschränkte σ-Struktur.

- Zwei rk-Umgebungen gehören demselben **Isomorphietyp** an, wenn sie isomorph zueinander sind.
- $\mathcal{T}_{rk}$ sei die Menge der Isomorphietypen der (rk)-Umgebungen $\mathcal{N}_{rk}(a)$ der einzelnen Elemente $a \in A$. Der (rk)**-Umgebungstyp** eines Elements $a \in A$ ist der Isomorphietyp seiner (rk)-Umgebung. *Beachte, dass die (rk)-Umgebung jedes Elements wegen der Beschränktheit von $\mathfrak{A}$ höchstens d^{rk} Elemente beinhaltet.*

- Für jedes $\tau \in \mathcal{T}_{rk}$ und $x \in A$ besagt $\mu_\tau(x)$, dass τ der (rk)-Umgebungstyp von x ist.
- Für jedes $\tau \in \mathcal{T}_{rk}$ wird ein **Repräsentant** der zu τ gehörigen (rk)-Umgebung gewählt. Unter den in dem Repräsentanten enthaltenen Elementen wird eine lineare Ordnung nach dem folgenden Schema festgelegt:
  - Sei a das Element im Zentrum des Repräsentanten. a erhält die Nummer 1.
  - Dann werden alle Elemente b mit $\delta(a, b) = 1$ durchnummeriert.
  - Anschließend werden alle Elemente b mit $\delta(a, b) = 2$ durchnummeriert.
  - usw.

  Auf diese Weise werden alle l-Umgebungen mit $l \leq rk$ geordnet. Das Nummerierungsschema gewährleiste dabei, dass die Nummerierung der l-Umgebung mit der Nummerierung der $(l + 1)$-Umgebung kompatibel ist.

Beispiel 4.11
Betrachtet wird ein Ausschnitt einer Graph-Struktur $\mathfrak{G}$. Die Nummerierung der 3-Umgebung eines Elements $a \in G$ könnte so aussehen:



Denkbar wäre eine spiralförmig verlaufende Nummerierung vom Zentrum nach außen. Man sieht insbesondere, dass sich die Nummerierung der i-Umgebung in der $(i + 1)$-Umgebung fortsetzt (hier $i = 0, 1, 2$).

Die Repräsentanten der Isomorphietypen liefern somit eine Art „Landkarte“ mit deren Hilfe sich alle (rk)-Umgebungstypen desselben Insomorphietypen beschreiben lassen.

Das im folgenden Beweis zentrale Konzept ist das der **r-Partition**. Die Idee, einem Variablentupel $(x_1, \ldots, x_k)$ eine r-Partition zuzuordnen, zielt drauf ab, die Möglichkeiten, das Tupel mit Elementen zu belegen, zu charakterisieren. Charakterisieren bedeutet in diesem Fall, anzugeben, ob die Elemente zueinander benachbart sind oder ob es einen großen Abstand zwischen ihnen gibt. Da es aber schwierig ist, sich die möglichen Nachbarschaften einer Variablen ohne konkreten Bezug

zu einer σ-Struktur vorzustellen, werden die grundlegenden Begriffe der r-Partition am Beispiel eines konkreten Tupels $\overline{a} = (a_1, \ldots, a_k) \in A^k$ eingeführt.

Sei $F = \{\alpha_2, \ldots, \alpha_m\} \in \{1, \ldots, d^{rk}\}^{m-1}$ eine Sequenz aus $(m-1)$ Zahlen. Die Schreibweise, um für das Tupel $\overline{a}$ auszudrücken, dass a_2 das Element mit der Nummer α_2 in der (rk)-Umgebung von a_1 ist, a_3 das Element mit der Nummer α_3 usw., sei:

$$\overline{a} = F\big|_{a_1} \tag{4.27}$$

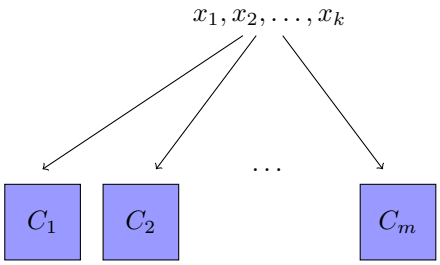
Beispiel 4.12
Betrachtet wird erneut der Ausschnitt der Graph-Struktur $\mathfrak{G}$ aus Beispiel 4.11. Sei $rk \geq 3$, sei $\overline{a} = (a, f, g)$ und $F = (6, 7)$. Die Angabe $\overline{a} = F\big|_a$ wäre somit für das Tupel $\overline{a}$ gültig.

Sei $\mathcal{F}_{rk}^m = \{1, \ldots, d\}^{m-1}$ die Menge aller möglichen F. Sei $\mathcal{F}_{rk} = \cup_{1 \leq m \leq k} \mathcal{F}_{rk}^m$ die Vereinigung all dieser Mengen von Tupeln aller Längen zwischen 0 und $(k-1)$ mit Einträgen aus dem Intervall $[1, d^{rk}]$.

Nun wird es etwas abstrakter. Sei $\overline{x} = (x_1, \ldots, x_k)$ ein Variablentupel. Die r-Partition des Tupels $\overline{x}$ sei die Menge $\{(C_1, F_1), \ldots, (C_m, F_m)\}$, wobei die Mengen F_i und C_i (mit $i = 1, \ldots, m$) die folgenden Eigenschaften besitzen:

- $\emptyset \neq C_i \subseteq \{x_1, \ldots, x_k\}$,
- $C_i \cap C_j = \emptyset$ für $i \neq j$,
- $\cup_{1 \leq i \leq m} C_i = \{x_1, \ldots, x_k\}$ sowie
- $F_i \in \mathcal{F}_{rk}^{|C_i|}$.

Jedem C_i wird ein Tupel $\overline{x}^i$ zugeordnet. $\overline{x}^i$ entsteht aus $\overline{x}$, indem diejenigen Elemente entfernt werden, die nicht in C_i enthalten sind. Weiterhin bezeichnet x_1^i das erste Element von $\overline{x}^i$, x_2^i das zweite usw. Anschaulich kann man sich die Mengen $C_1, \ldots, C_m$ vorstellen wie Schubladen, auf die die k Elemente von $\overline{x}$ verteilt werden, ohne dass eine Schublade leer ausgehen darf:



Die einzelnen Mengen F_i beinhalten, wie oben beschrieben die Zahlen, die den C_i zugeordnet werden.

Beispiel 4.13
Sei $\mathcal{F}_{rk}^{|C_i|} = \{1, 2, 3, 4, 5\}^{|C_i|-1}$. Sei $\overline{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. Sei $m = 3$ die Anzahl der „Schubladen“ C_1, C_2, C_3. Eine mögliche r-Partition $C = \{(C_1, F_1), (C_2, F_2), (C_3, F_3)\}$ für $\overline{x}$ wäre

$C_1 = \{x_3, x_4, x_5\}$ $\quad$ $F_1 = (2, 5)$
$C_2 = \{x_1, x_2, x_7\}$ $\quad$ $F_2 = (3, 5)$
$C_3 = \{x_6\}$ $\quad$ $F_3 = ()$
und somit
$\overline{x}^1 = (x_1^1, x_2^1, x_3^1) = (x_3, x_4, x_5)$
$\overline{x}^2 = (x_1^2, x_2^2, x_3^2) = (x_1, x_2, x_7)$
$\overline{x}^1 = x_1^3 = x_6$

Ohne Weiteres hat eine r-Partition also keine Aussagekraft bezüglich der Lage der Elemente im Universum einer σ-Struktur, die das Variablentupel belegen. Die Bedeutung wird ihr erst im Kontext mit einer FO-Formel verliehen. Diese FO-Formel sei $Div_r^C(\overline{x})$. Für eine gegebene r-Partition $C = \{(C_1, F_1), \ldots, (C_m, F_m)\}$ und das Variablentupel $\overline{x} = (x_1, \ldots, x_k)$ sei $Div_r^C(\overline{x})$ eine Konjunktion von

- FO-Formeln, die aussagen, dass für alle $1 \leq i \neq j \leq m$ gilt: $N_r^{\mathfrak{A}}(\overline{x}^i) \cap N_r^{\mathfrak{A}}(\overline{x}^j) = \emptyset$ und von
- FO-Formeln, die ausdrücken, dass $\overline{x}^i$ gebunden um x_1^i ist, d.h. $\bigwedge_{(C_i,F_i)\in C} \overline{x}^i = F_i\big|_{x_1^i}$.

Mit anderen Worten, $Div_r^C(\overline{x})$ ist eine FO-Formel, die so konstruiert ist, dass für alle Tupel $\overline{a} \in A^k$ und alle σ-Strukturen $\mathfrak{A}$ gilt:

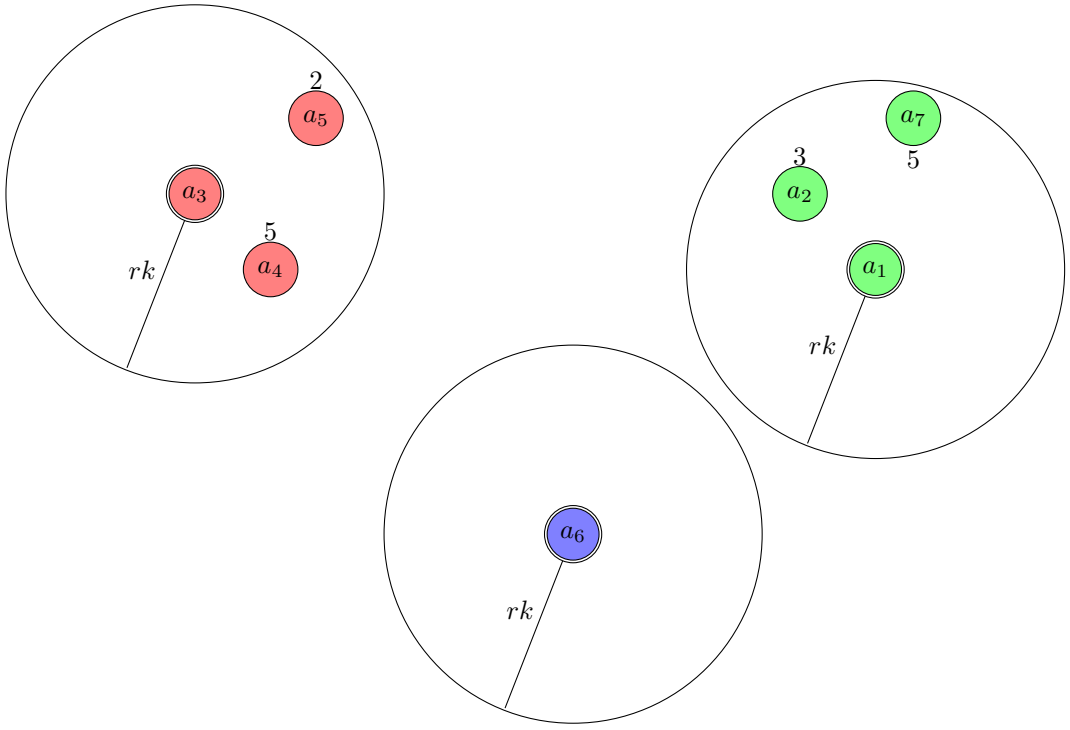
$$\begin{aligned} \mathfrak{A} \models Div_r^C[\overline{a}] \iff & \overline{a} \text{ verhält sich gemäß der Partition } C \text{ d.h.,} \\ & N_r(\overline{a}^i) \cap N_r(\overline{a}^j) = \emptyset \text{ f.a. } 1 \leq i \neq j \leq m \\ & \text{und} \bigwedge_{(C_i,F_i)\in C} \overline{a}^i = F_i\big|_{a_1^i} \end{aligned} \tag{4.28}$$

wobei $\overline{a}^i$ bedeutet, dass die Elemente von $\overline{x}$, die gemäß C zu C_i gehören, durch die entsprechenden Elemente aus $\overline{a}$ belegt wurden.

Beispiel 4.14
Sei $\overline{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$, sei $\overline{a} = (a_1, a_2, a_3, a_4, a_5, a_6, a_7)$.
Wenn $C_1 = (x_3, x_4, x_5) = \overline{x}^1$, so ist $\overline{a}^1 = (a_3, a_4, a_5)$.

Analog ist die Schreibweise a_1^i zu verstehen. Man beachte, dass $Div_r^C(\overline{x})$ impliziert, dass $\overline{x}^i$ gebunden um x_1^i ist. Anschaulich kann man sich die Lage eines Tupels $\overline{a}$, das $Div_r^C(\overline{x})$ erfüllt, vorstellen wie eine Inselgruppe (vgl. auch Beispiel 4.13):

Beispiel 4.15 („Inselgruppe“)
Betrachte einen Ausschnitt von $\mathfrak{A}$.



$Div_r^C(\overline{x})$ drückt aus, dass man sich die Verteilung der Elemente von $\overline{a}$ in $\mathfrak{A}$ vorstellen kann wie eine Inselgruppe mit $|C|$-vielen Inseln vom Radius rk.

Mit dem nötigen Rüstzeug kann nun zum Kern dieses Kapitels vorgedrungen werden. Die zentrale Aussage, die von Kazana und Segoufin bewiesen wird, lässt sich folgendermaßen formulieren:

Satz 4.16
Es existiert ein Algorithmus, der das Aufzählungsproblem für alle $d \in \mathbb{N}$, alle durch d beschränkten Strukturen $\mathfrak{A}$ und alle FO-Formeln ϕ mit Platzbedarf linear in $\|\mathfrak{A}\|$ und der folgenden Laufzeit löst:

- Vorverarbeitungsphase: $2^{2^{2^{\mathcal{O}(|\phi|)}}} \cdot \|\mathfrak{A}\|$
- Verzögerung in der Ausgabephase: $2^{2^{2^{|\phi|}}}$, also Gesamtlaufzeit der Berechnung der Lösungen in $2^{2^{2^{|\phi|}}} \cdot |\phi(\mathfrak{A})|$

Das bedeutet somit, dass das Aufzählungsproblem für Strukturen von beschränktem Grad in Constant-Delay$_{lin}$ liegt.

Bevor Satz 4.16 bewiesen werden kann, wird der im folgenden Lemma vorgestellte Sachverhalt über die Darstellbarkeit von FO-Formeln benötigt:

Lemma 4.17
Für eine gegebene Struktur $\mathfrak{A}$ ist jede beliebige FO-Formel $\phi(\overline{x})$ äquivalent auf $\mathfrak{A}$ zu einer Formel der Form:

$$\bigvee_{C \in C_r(x_1,\ldots,x_k)} \left(Div_r^C(\overline{x}) \wedge \bigvee_{(\tau_1,\ldots,\tau_{|C|}) \in S_c} \bigwedge_{i \leq |C|} \mu_{\tau_i}(x_1^i) \right) \tag{4.29}$$

dabei ist $C_r(x_1, \ldots, x_k)$ die Menge aller r-Partitionen von $(x_1, \ldots, x_k)$, $S_C \subseteq (\tau_{rk})^{|C|}$ ist endlich und $r = 7^{qr(\phi)}$.

Beweis. Sei $\mathfrak{A}$ eine beliebige Struktur. Sei $\phi(\overline{x})$ eine FO-Formel mit k freien Variablen und sei $r = 7^{qr(\phi)}$. Sei $C_r(\overline{x})$ die Menge aller Partitionen $C = ((C_1, F_1), \ldots, (C_m, F_m))$ von $\overline{x}$ mit $C_i = \{x_1^i, \ldots, x_{|C_i|}^i\}$. Sei $\phi'(\overline{x})$ von der Form (4.29).

Zu zeigen: Für alle $\overline{a} \in A^k$ gilt:

$$\mathfrak{A} \models \phi[\overline{a}] \Longleftrightarrow \mathfrak{A} \models \phi'[\overline{a}] \tag{4.30}$$

„⇒“:
Berücksichtigt man alle möglichen r-Partitionen von $\overline{x}$, so gilt, dass es unter ihnen stets eine gibt, sodass $Div_r^C(\overline{x})$ mit $\phi(\overline{x})$ vereinbar ist. D.h., es gilt in jedem Fall dass $\phi(\overline{x})$ äquivalent ist zu

$$\bigvee_{C \in C_r(\overline{x})} \big(Div_r^C(\overline{x}) \wedge \phi(\overline{x}) \big) \tag{4.31}$$

Sei $\overline{a}$ eine Lösung für $\phi(\overline{x})$, d.h. $\mathfrak{A} \models \phi[\overline{a}]$. Für genau eine r-Partition C von $\overline{x}$ gilt dann $\mathfrak{A} \models \big(Div_r^C[\overline{a}] \wedge \phi[\overline{a}] \big)$, nämlich genau für diejenige r-Partition C, die die Lage der Elemente von $\overline{a}$ in $\mathfrak{A}$ hinsichtlich $\phi(\overline{a})$ richtig beschreibt. Die Aussage von $Div_r^C(\overline{x})$ ist, wie oben beschrieben, dass für alle C_i aus $(C_i, F_i) \in C$ gilt, dass die enthaltenen Elemente $\overline{a}^i$ jeweils um a_1^i gebunden sind, also $\overline{a}^i$ vollständig in der (rk)-Umgebung von a_1^i erhalten ist, vgl. Beispiel 4.15. Sei $m = |C|$. In Anlehnung an das Beispiel bedeutet das, dass die zu C gehörige „Inselgruppe“ m Mitglieder hat. Für $1 \leq i \leq m$ sei durch τ_i der (rk)-Umgebungstyp von a_1^i benannt. Sei S_C die Menge aller Tupel $(\tau_1, \ldots, \tau_m)$ von (rk)-Umgebungstypen für alle $\overline{a}$, für die gilt: $\mathfrak{A} \models \big(Div_r^C[\overline{a}] \wedge \phi[\overline{a}] \big)$. Die Formel ϕ trägt die Information, wie die Elemente, „über die sie spricht“, zueinander liegen und wie ihre Umgebung aussieht. Genau dies vermittelt (4.29). Somit ist ϕ ohne Informationsverlust

in die Form (4.29) transformierbar.

„⇐“:
Zum Beweis der Implikation ist das folgende Lemma hilfreich:

Lemma 4.18
Für alle Tupel $\overline{a} \in A^k$ gilt: Die r-Partition C induziert die Substruktur $\mathcal{N}_r^{\mathfrak{A}}(\overline{a})$, so dass $\mathcal{N}_r^{\mathfrak{A}}(\overline{a}) \models \phi'[\overline{a}]$.

Die Implikation folgt unter Verwendung von Lemma 4.18 ohne Weiteres aus Satz 4.7 mit:

$$\mathfrak{A} \models \phi'[\overline{a}] \Longrightarrow \mathcal{N}_r^{\mathfrak{A}}(\overline{a}) \models \phi'[\overline{a}] \Longrightarrow \mathcal{N}_r^{\mathfrak{A}}(\overline{a}) \models \phi[\overline{a}] \Longrightarrow \mathfrak{A} \models \phi[\overline{a}] \tag{4.32}$$

□

Beweis von Lemma 4.18. Sei $\overline{a}$ ein Tupel, sodass

$$\mathfrak{A} \models \phi'[\overline{a}] \tag{4.33}$$

D.h.

1. Es gibt ein $C \in C_r(\overline{x})$, so dass $\mathfrak{A} \models Div_r^C[\overline{a}]$.
2. Es gibt eine Sequenz $(\tau_1, \ldots, \tau_{|C|}) \in \mathcal{T}_{rk}^{|C|}$, so dass für alle $i \leq |C|$ gilt: $\mu_{\tau_i}(a_1^i)$ und $\mathcal{N}_{rk}^{\mathfrak{A}}(a_1^i) \cong \tau_i$.

Weiterhin gilt $\mathcal{N}_{rk}^{\mathfrak{A}}(a_1^i) \supseteq \mathcal{N}_r^{\mathfrak{A}}(\overline{a}^i)$. Durch die disjunkte Vereinigung der einzelnen r-Umgebungen $\mathcal{N}_r^{\mathfrak{A}}(\overline{a}^i)$ wird $\mathcal{N}_r^{\mathfrak{A}}(\overline{a})$ induziert. □

Nun kann der Beweis von Satz 4.16 geführt werden.

Beweis. Sei $\phi(\overline{x})$ eine FO-Formel mit k freien Variablen und sei $\mathfrak{A}$ eine durch d beschränkte Struktur. O.B.d.A. kann angenommen werden, dass das Universum von $\mathfrak{A}$ linear geordnet ist[1]. Sei $r = 7^{qr(\phi)}$. $\phi(\overline{x})$ ist äquivalent zu einer Formel von der in 4.29 beschriebenen Form.

Der Beweis wird in zwei Teile eingeteilt: in die Betrachtung der Vorverarbeitungs- und in die der Ausgabephase.

Teil 1: Vorverarbeitungsphase
Zur Vorverarbeitungsphase zählt in jedem Fall die Umwandlung von $\phi(\overline{x})$ in die Form (4.29). Die dazu nötigen Informationen werden in vier Schritten erarbeitet. Weiterhin werden die (rk)-Umgebungen aller Elemente aus A ermittelt. Die Ermittlung einer Laufzeit, die tatsächlich zu CONSTANT-DELAY$_{lin}$ passt, stützt sich auf Satz 4.8. Im Einzelnen setzt sich die Vorverarbeitungsphase aus den folgenden Schritten zusammen:

Schritt 1: Für jedes Element $a \in A$ berechne die Menge $B_i = \{b : \delta(a, b) = i\}$ für alle $i \leq (rk)$.
Behauptung:
Es gibt einen Algorithmus, der dies in Zeit $\mathcal{O}(\|\mathfrak{A}\|)$ vermag.

Beweis. Der Beweis erfolgt per Induktion über die Distanz i.

$i = 1$:
Die Induktionsbasis umfasst die Erstellung des Gaifman-Graphen von $\mathfrak{A}$. Initialisiere dazu $|A|$ Mengen B_1^a, die für jedes Element $a \in A$ die jeweiligen Nachbarelemente speichern. Für jedes Tupel, das in $\mathfrak{A}$ enthalten ist, gehe die Elemente des Tupels nacheinander durch und füge dabei für jedes Element a im Tupel die übrigen Elemente zu B_i^a hinzu. Da $\mathfrak{A}$ einen durch d beschränkten Grad hat, hat jedes Tupel höchstens die Länge d. Weiterhin gibt es

[1] Zumindest über die Benennung der einzelnen Elemente kann stets eine lexikographische Ordnung erreicht werden.

höchstens $(d! \cdot \|\mathfrak{A}\|)$-viele Tupel. Die Induktionsbasis benötigt also Zeit linear in $\|\mathfrak{A}\|$. Sie liefert alle Informationen, um den Gaifman-Graphen von $\mathfrak{A}$ zu erstellen. Dies geht ebenso in Linearzeit vonstatten, da aus den einzelnen Listen B_1^a lediglich noch die für einen Graphen übliche Datenstruktur erstellt werden muss. Für die Operationen in den folgenden Schritten liegt nun stets der Gaifman-Graph vor.

Induktionsannahme:
Für alle Elemente $a \in A$ wurde in Zeit linear in $\|\mathfrak{A}\|$ die Mengen B_i^a erstellt, $i < rk$.

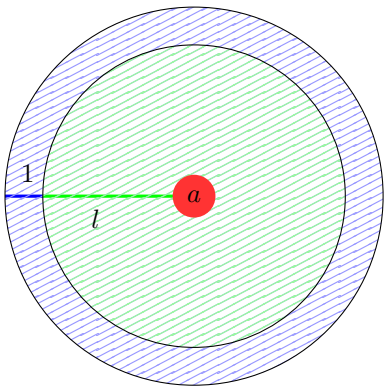
$i \to i+1$:
Initialisiere für alle $a \in A$ die Mengen B_{i+1}^a. Für alle $a \in A$ durchlaufe die Elemente b aus den Mengen B_i^a und füge für jedes Element b die im Gaifman-Graphen benachbarten Knoten zu B_{i+1}^a hinzu, sofern sie sich nicht in der Menge B_i^a befinden. Da $i+1 \leq rk$, befinden sich in den Mengen B_i^a und B_{i+1}^a jeweils höchstens d^{rk} Elemente. Folglich benötigt der Induktionsschritt wiederum die Zeit $\mathcal{O}(\|\mathfrak{A}\|)$. Für alle Distanzen $i+1 \leq rk$ wird also die Zeit $rk \cdot \mathcal{O}(\|\mathfrak{A}\|)$ benötigt. Der gesamte Schritt 1 erfolgt somit in Zeit $\mathcal{O}(\|\mathfrak{A}\|)$. □

Schritt 2: Für jeden Knoten a des Graphen wird sein (rk)-Umgebungstyp ermittelt und ein Zeiger von a zu dem i-ten Element des (rk)-Umgebungstyps für alle $i \leq d^{rk}$ gesetzt. Die Laufzeit des Verfahrens wird per Induktion über den Radius l der Umgebung berechnet:

$l = 0$:
Der Induktionsanfang liegt bei einer 0-Umgebung. Alle Elemente teilen denselben Umgebungstyp und der Zeiger jedes Elements zeigt nur auf das Element selbst. Die benötigte Zeit ist folglich $\mathcal{O}(\|\mathfrak{A}|)$.

Induktionsannahme
Die Induktionsannahme ist, dass in Zeit $\mathcal{O}(\|\mathfrak{A}|)$ bereits gelungen ist, die l-Umgebung aller Elemente zu ermitteln.

$l = l+1$:
Gehe alle Elemente $a \in A$ nacheinander durch. Basierend auf den Ermittlungen aus Schritt 1 stehen für jedes a die Elemente b mit $\delta(a,b) = l+1$ zur Verfügung. Die Frage ist nun, in welcher Reihenfolge diese Elemente geordnet sind. Dazu werden alle möglichen Reihenfolgen ausprobiert und jeweils mit den eingangs festgelegten Repräsentanten der $(l+1)$-Umgebung verglichen. Äußerst grob abgeschätzt gibt es höchstens d^{l+1} Elemente im Abstand $(l+1)$ (in Wirklichkeit sind es i.d.R. weit weniger). Da $l < rk$, ist folglich die Anzahl der möglichen linearen Ordnungen, die ausprobiert werden müssen, durch $d^{rk}!$ beschränkt. Die folgende Abbildung verdeutlicht die Überlegung:



Die schon bekannte l-Umgebung ist grün schraffiert, die Elemente im Anstand $(l+1)$ werden

durch die blaue Schraffur angedeutet.

Für jede Ordnung wird für alle (rk)-Umgebungstypen (das sind $|\mathcal{T}_{rk}|$-viele) geprüft, ob zwischen beiden ein Isomorphismus besteht. Dies erfordert lediglich den direkten Vergleich der Elemente in den (rk)-Umgebungen, also genau so viele Schritte, wie die (rk)-Umgebung groß ist. Sei $s(r,k,d)$ die maximale Größe einer (rk)-Umgebung. Zweifellos ist $s(r,k,d) = \mathcal{O}(d^{rk|\sigma|})$. Die Anzahl der unterschiedlichen (rk)-Umgebungstypen $|\mathcal{T}_{rk}|$ ist exponentiell in der Größe einer (rk)-Umgebung und darum abschätzbar durch $\mathcal{O}(2^{s(r,k,d)})$. Dies ist auch durch die Überlegungen im Beweis von Satz 4.8 ersichtlich, soll aber an dieser Stelle noch auf einem alternativem Weg gezeigt werden: Angenommen, die rk-Umgebung hätte d^{rk} Elemente und ihr der Gaifman-Graph folglich höchstens d^{rk} Kanten, d.h., jedes Element hätte d Nachbarn. Es gibt höchstens $d^{rk}! \leq (d^{rk})^{d^{rk}}$ verschiedene Möglichkeiten, die Kanten in der rk-Umgebung anzuordnen. Falls nicht jedes Element d Nachbarn hätte, wäre die Zahl der Möglichkeiten entsprechend reduziert. Höchstens können mit dieser Überlegung aber $\mathcal{O}((d^{rk})^{d^{rk}})$ verschiedenen Fälle und somit auch $\mathcal{O}((d^{rk})^{d^{rk}})$ verschiedene (rk)-Umgebungstypen auftreten. Die Abschätzung von $|\mathcal{T}_{rk}|$ durch $\mathcal{O}(2^{s(r,k,d)})$ ist folglich richtig.

Insgesamt benötigt Schritt 2 die Zeit:

$$\mathcal{O}(\underbrace{|\tau_{rk}|}_{\mathcal{O}(2^{d^{rk|\sigma|}})} (d^{rk})!\, d^{rk|\sigma|}) \tag{4.34}$$

wegen $r = 7^{qr(\phi)}$ ist sie dreifach exponentiell in $|\phi|$ wie gefordert.

Schritt 3: Nun werden die für ϕ auf $\mathfrak{A}$ relevanten (rk)-Umgebungstypen errechnet. Wähle eine r-Partition $C \in C_r(\overline{x})$ und eine Sequenz $(\tau_1, \ldots, \tau_m) \in {\mathcal{T}_{rk}}^m$ Diese Sequenz ist relevant für C, falls

$$\mathfrak{A} \models \exists \overline{x} \left(Div_r^C(\overline{x}) \wedge \bigwedge_j \mu_{\tau_j}(x_1^j) \right) \wedge \phi(\overline{x}) \tag{4.35}$$

Ansonsten gäbe es, angelehnt an Beispiel 4.15, keine „Einteilung in Inseln“, auf denen die (rk)-Umgebungstypen $\tau_1, \ldots, \tau_m$ zu finden sind. Um $\mu_{\tau_j}(x_1^j)$ zu testen, d.h., um zu prüfen, ob τ_j der (rk)-Umgebungstyp von x_1^j ist oder nicht, wird konstante Zeit benötigt, da in Schritt 2 die (rk)-Umgebungstypen aller Elemente bestimmt worden sind und man für jedes Element lediglich nachzuschlagen braucht, ob der jeweilige (rk)-Umgebungstyp τ_j ist oder nicht. Ähnlich verhält es sich mit $Div_r^C(\overline{x})$. Um für eine Belegung $\overline{a}$ von $Div_r^C(\overline{x})$ zu kontrollieren, ob $\mathfrak{A} \models Div_r^C[\overline{a}]$, also ob sich $\overline{a}$ gemäß C verhält, muss für alle C_i aus $(C_i, F_i) \in C$ getestet werden, ob

1. $\overline{a}^i = F_i\big|_{a_1^i}$ und ob
2. $N_r^{\mathfrak{A}}(\overline{a}^i) \cap N_r^{\mathfrak{A}}(\overline{a}^k) = \emptyset$.

Punkt 1 ist zu überprüfen, indem für das Element a_1^i für alle $l \leq rk$ und alle $a_j^i \in \overline{a}^i$ getestet wird, ob für die in Schritt 1 erstellten Mengen gilt: $a_j^i \in B_l^a$. Dies geschieht in Zeit $\mathcal{O}(rk \cdot |B_{rk}^a| \cdot |\overline{a}^i|) = \mathcal{O}(d^{rk} \cdot rk^2)$. Die benötigte Zeit ist somit konstant. Punkt 2 ist überprüfbar, indem für alle $i \leq r$ und alle $a_j^i \in \overline{a}^i$ getestet wird, ob für alle $l \leq r$ und $a_{j'}^k \in \overline{a}^i$ die Mengen $B_i^{a_j^i}$ und $B_l^{a_{j'}^i}$ gleiche Elemente enthalten. Dies ist in Zeit $\mathcal{O}(d^{rk} \cdot (rk)^2)$ und somit wieder in konstanter Zeit möglich. $\mu_{\tau_j}(x_1^j)$ und $Div_r^C(\overline{x})$ können somit wie ein- bzw. k-stellige Relationssymbole betrachtet werden, womit $Div_r^C(\overline{x}) \wedge \bigwedge_j \mu_{\tau_j}(x_1^j)$ wie eine Formel der Größe $\mathcal{O}(|\overline{x}|)$ betrachtet werden kann. Mithilfe von Satz 4.8 sieht man, dass in einer Zeit linear in $\|\mathfrak{A}\|$ und dreifach exponentiell in $|\phi|$ getestet werden kann, ob eine Sequenz $(\tau_1, \ldots, \tau_m)$ relevant für C ist. Man tut dies für alle möglichen C und muss wegen $m \leq k$ jeweils höchstens $|\mathcal{T}_{rk}|^k$ Fälle betrachten. Die Anzahl der möglichen C ist die Anzahl

von möglichen Einteilungen von k Variablen in m Sequenzen multipliziert mit $|\mathcal{F}_{rk}|^k$, was wiederum dreifach exponentiell in $|\phi|$ ist. Für jedes C wird eine Liste aller für C relevanten Sequenzen gespeichert. Eine r-Partition heißt relevant, falls die zugehörige Liste nicht leer ist.

Schritt 4: Im letzten Schritt werden die Elemente aus A gemäß ihres (rk)-Umgebungstypen $\tau \in \mathcal{T}_{rk}$ gruppiert und je ein Zeiger von einem Element zu seinem Nachfolger gesetzt. Dazu werden die Elemente aus A gemäß ihrer ursprünglichen Ordnung durchgegangen und anhand der in Schritt 2 gefundenen Informationen jeweils in einer geeigneten Liste gespeichert. Dies ist in Zeit $\mathcal{O}(\|\mathfrak{A}\|)$ möglich (vgl. die in [6] vorgestellte Methode, auf die an dieser Stelle nicht näher eingegangen werden soll).

Man beachte, dass der Platzbedarf von allen Schritten linear in $\|\mathfrak{A}\|$ ist. Damit kann die Vorverarbeitungsphase die zeitliche Anforderung erfüllen.
Für die weiteren Überlegungen ist das folgende Lemma wichtig:

Lemma 4.19
Falls zwei Probleme R und R' beide in CONSTANT-DELAY$_{lin}$ liegen und es außerdem eine lineare Ordnung $<$ gibt, sodass die Lösungen von R und R' in dieser Ordnung ausgegeben werden, dann liegt das Problem $R \cup R'$ ebenfalls in CONSTANT-DELAY$_{lin}$ und seine Lösungen können durch $<$ geordnet ausgegeben werden.

Beweis. Der Beweis basiert lediglich auf der geordneten Vereinigung zweier geordneter Listen. □

Teil 2: Ausgabephase
Sei $C = ((C_1, F_1), \ldots, (C_m, F_m))$ eine relevante r-Partition und $(\tau_1, \ldots, \tau_m) \in {\mathcal{T}_{rk}}^m$ eine für C relevante Sequenz. Die Ermittlung der Lösungen für $\phi(\overline{x})$ erfolgt per Induktion über die Klassen m in C.

$\underline{m = 1}$: Dank der Vorbereitung in der Vorverarbeitungsphase stehen in konstanter Zeit alle Elemente $a_1^1 \in A$ zur Verfügung, deren (rk)-Umgebungstyp τ_1 ist. Die Lösungen der Induktionsbasis sind $\overline{a}^1 = F_1\big|_{a_1^1}$.

Induktionsannahme: Alle Lösungen $\overline{b}$ für $\Psi(\overline{x}) = Div_r^{C'}(\overline{x}) \wedge \bigwedge_j \mu_{\tau_j}(x_1^j)$ mit $C' = C \backslash (C_m, F_m)$ werden in konstanter Zeit gefunden.

$\underline{m - 1 \to m}$:
In konstanter Zeit werden alle Elemente a_1^m aufgezählt, deren (rk)-Umgebungstyp τ_m ist. Für jede laut Induktionsannahme gefundene Lösung $\overline{b}$ wird nun geprüft, ob $N_{rk}^{\mathfrak{A}}(a_1^m) \cap N_r^{\mathfrak{A}}(\overline{b}) = \emptyset$. Falls dies der Fall ist, ist $(\overline{b}, \overline{a}^m)$ eine Lösung von $\phi(\overline{x})$. Falls nicht, wird die nächste Lösung $\overline{b}$ untersucht. Beachte: Es gibt mindestens eine Lösung $(\overline{b}, \overline{a}^m)$, da die Sequenz $(\tau_1, \ldots, \tau_m)$ relevant ist für C. Die Zeit zwischen den Funden zweier Lösungen ist konstant, was folgende Überlegung zeigt: $N_{rk}^{\mathfrak{A}}(a_1^m)$ enthält höchstens d^{rk} Elemente. Wegen $|\overline{b}| \leq k$ kann somit kann der Fall, dass sich $N_{rk}^{\mathfrak{A}}(a_1^m)$ und $N_r^{\mathfrak{A}}(\overline{b})$ überschneiden höchstens $(d^{rk})^k = d^{rk^2}$ mal auftreten. Somit ist die Verzögerung, mit der Lösungen ausgegeben werden, konstant. Der Vorgang wird für alle für C relevanten Sequenzen $(\tau_1, \ldots, \tau_m) \in {\mathcal{T}_{rk}}^m$ wiederholt. Ihre Anzahl ist durch $2^{2^{2^{\mathcal{O}(|\phi|)}}}$ begrenzt. Laut Lemma 4.19 kann das mehrfache Vorkommen derselben Lösung dabei vermieden werden. Weiterhin müssen alle relevanten r-Partitionen berücksichtigt werden. Auch ihre Zahl ist in jedem Fall begrenzt durch $2^{2^{2^{\mathcal{O}(|\phi|)}}}$. Alle Lösungen sind unabhängig voneinander und können somit sequenziell ermittelt werden. Insgesamt benötigt die Ausgabephase Zeit dreifach exponentiell in der Länge von ϕ multipliziert mit der Anzahl der Lösungstupel von ϕ, nämlich $2^{2^{2^{\mathcal{O}(|\phi|)}}} \cdot |\phi(\mathfrak{A})|$. Der Platzbedarf ist wiederum linear in $\|\mathfrak{A}\|$ Die Ausgabephase erfüllt somit die Forderung aus Satz 4.16. □

Somit ist der Beweis von Satz 4.16 abgeschlossen und damit eine zweite Beweisstrategie präsentiert.

5 Zusammenfassung und Ausblick

5.1 Kurze Zusammenfassung beider Beweise

In dieser Arbeit wurden zwei verschiedene Strategien präsentiert, um zu beweisen, dass das Auswertungsproblem auf Strukturen von beschränktem Grad in der Klasse CONSTANT-DELAY$_{lin}$ liegt. Auf zwei verschiedene Arten wurde gezeigt, dass es einen Algorithmus gibt, der das Auswertungsproblem auf Strukturen von beschränktem Grad in der Zeit $f(k) \cdot (n + m)$ löst. Dazu wurden das Konzept des Auswertungsproblems sowie das der Klasse CONSTANT-DELAY$_{lin}$ formalisiert.

Im Rahmen der Beweisstrategie nach DURAND und GRANDJEAN [2] wurde der Begriff der bijektiven Struktur vorgestellt und gezeigt, dass das Auswertungsproblem für bijektive Strukturen in CONSTANT-DELAY$_{lin}$ liegt. Kern der Strategie war die Umwandlung einer Struktur von beschränktem Grad in eine bijektive Struktur, welche anhand eines Beispiels verdeutlicht wurde.

Der zweite Beweis nach KAZANA und SEGOUFIN [3] bediente sich der Gaifman-Lokalität für FO-Formeln. Es wurde ein Algorithmus angegeben, um das Auswertungsproblem auf Strukturen von beschränktem Grad ohne Umweg über die Umwandlung in eine bijektive Struktur zu lösen. Im Zentrum der Überlegungen stand der Gaifman-Graph, in dessen Zusammenhang vom Konzept der r-Partition und von dem der r-Umgebung Gebrauch gemacht wurde.

5.2 Vergleich beider Beweise

Die beiden vorgestellten Beweisstrategien sind in ihrer Grundidee und Darstellung sehr unterschiedlich. Folgende Aussagen beziehen sich auf die Originalarbeiten [2, 3].

Durand und Grandjean legen einen sehr ausführlichen und geradlinigen Beweis vor. Er ist nicht schwer verständlich und bietet sogar einige Beispiele, die dem Leser den Zugang zu den vorgestellten Konzepten erleichtern. Die Strategie, den Beweis für das Aufzählungsproblem zunächst auf einer anderen Klasse von Strukturen als der eigentlich gewünschten zu betrachten und einen Algorithmus anzugeben, der die gewünschte Struktur aus der betrachteten konstruiert, wirkt allerdings recht umständlich und aufwendig.

Viel eleganter wirkt dagegen der Beweis von Kazana und Segoufin. Dieser ist sehr kompakt dargestellt und ermöglicht mithilfe ausgeklügelter Begriffsdefinitionen eine geradlinige Beweisführung, die sich unmittelbar der Beschränktheit der Struktur bedient. Die Eleganz des Beweises hat allerdings seinen Preis: Die Definition der Begriffe erfolgt ohne Beispiel und birgt die Gefahr von Doppeldeutigkeiten. Es ist schwierig, angesichts der hohen Argumentationsdichte den Überblick zu behalten.

5.3 Ausblick

In Zukunft könnte auf eine offene Frage, die Kazana und Segoufin in ihren Ausführungen aufwerfen, eingegangen werden. Es handelt sich um die Frage, ob für Auswertungsprobleme, die sich in der Zeit $f(k) \cdot (n + m)$ lösen lassen, auch stets ein der Klasse CONSTANT-DELAY$_{lin}$ entsprechender Algorithmus existiert. Die Vermutung der Autoren ist, dass dies nicht der Fall ist. Die Aufgabe wäre, diese Einschätzung rigoros zu bestätigen oder zu widerlegen.

Weiterhin könnte man über gänzlich andere Beweisstrategien nachdenken. Eine Möglichkeit, eine weitere Beweisstrategie zu finden, basiert auf dem Begriff der Hanf-Normalform (HNF). Eine Formel ist in HNF, wenn sie eine boolesche Kombination von Hanf-Formeln ist. Kurz gesagt ist eine Hanf Formel eine FO-Formel, die impliziert, dass es für einen Radius $r > 0$ eine bestimmte

Anzahl von r-Umgebungen gibt, die einen vorgegebenen Isomorphietyp teilen. In [7] wird ein Algorithmus unter Angabe seines Zeitaufwands präsentiert, um eine beliebige FO-Formel in HNF zu transformieren. Ausgehend von der HNF könnte analog zu den Überlegungen von KAZANA und SEGOUFIN ein Algorithmus zur Lösung des Auswertungsproblems erarbeitet werden.

Literaturverzeichnis

[1] Nicole Schweikardt. *Vorlesungsskript „Logik in der Informatik“*. Goethe-Universität Frankfurt am Main, `http://www.tks.informatik.uni-frankfurt.de/teaching/ws13/loginf`, Wintersemester 2013/2014. Zuletzt aufgerufen am 1.9.2014.

[2] Arnaud Durand and Etienne Grandjean. First-order queries on structures of bounded degree are computable with constant delay. *ACM Trans. Comput. Logic*, 8(4), August 2007.

[3] W. Kazana and L. Segoufin. First-order query evaluation on structures of bounded degree. *Logical Methods in Computer Science*, 7(2.20), June 2011.

[4] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[5] Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, 130(1-3):3–31, 2004.

[6] Etienne Grandjean. Sorting, linear time and the satisfiability problem. *Ann. Math. Artif. Intell.*, (16):183–236, 1996.

[7] Lucas Heimberg, Dietrich Kuske, and Nicole Schweikardt. An optimal gaifman normal form construction for structures of bounded degree. *Proc. 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, 0:63–72, June 25-June 28 2013.

Dank

Mein Dank gilt Nicole Schweikardt, die mir in ihren Vorlesungen die Welt der formalen Logik eröffnet hat. Sie hat sich sich meinen zahllosen Fragen und Nachfragen stets mit großer Geduld gewidmet und geizte nie mit exakten und anschaulichen Erklärungen.

Vielen Dank auch an Jutta Nadland für die vielen netten Gespräche und die moralische Unterstützung.

Selbstständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen Quellen oder Hilfsmittel als die in dieser Arbeit angegebenen verwendet habe.

Frankfurt am Main, den