Grundlagen der Anfrageverarbeitung beim relationalen Datenaustausch

André Hernich

Humboldt-Universität zu Berlin
hernich@informatik.hu-berlin.de

Abstract: Relationaler Datenaustausch behandelt die Übersetzung relationaler Datenbanken eines Schemas in ein anderes Schema. Eine grundlegende Frage ist, wie Anfragen an das Zielschema beantwortet werden können. Während man sich in dieser Frage bei so genannten monotonen Anfragen einig ist, hat sich deren Beantwortung für nicht-monotone Anfragen als weitaus schwieriger herausgestellt. Diese Arbeit gibt einen Überblick über die Grundlagen der Anfrageverarbeitung im relationalen Datenaustausch mit Schwerpunkt auf nicht-monotone Anfragen.

1 Einleitung

Beim relationalen Datenaustausch geht es um die Übersetzung relationaler Datenbanken eines Schemas in ein anderes Schema. Dies ist eines von vielen Problemen, die bei der Informationsintegration anfallen und unterliegt Anwendungen wie der Datenrestrukturierung und dem Austausch von Daten zwischen unabhängig voneinander entwickelten Anwendungen. Werkzeuge für diese Aufgabe existieren bereits seit längerer Zeit. Neuere Werkzeuge wie *Clio* nehmen solche Übersetzungen typischerweise automatisch basierend auf einem *Schema-Mapping* – einer *deklarativen* Spezifikation der Beziehungen zwischen Quell- und Zieldaten – vor. Aufbauend auf den Erfahrungen mit Clio haben sich Fagin, Kolaitis, Miller und Popa in der wegweisenden Arbeit [FKMP05] genauer mit grundlegenden und algorithmischen Fragestellungen zum relationalen Datenaustausch auseinandergesetzt. Mittlerweile nimmt die Forschung auf diesem Gebiet einen wichtigen Platz in der Datenbanktheorie ein und konzentriert sich auch auf Themen wie z.B. die Übersetzung von XML-Daten, Mehr-Parteien-Datenaustausch und die Manipulation (Komposition, Inversion u.a.) von Schema-Mappings selbst. Die Arbeiten [Kol05, Bar09] und das kürzlich erschienene Buch [ABLM10] geben einen schönen Überblick.

Formal besteht ein Schema-Mapping aus einem Quellschema, einem Zielschema und einer endlichen Menge von Aussagen, die die Beziehungen zwischen Quell- und Zieldaten herstellen. In der Literatur betrachtet man häufig Schema-Mappings, bei denen die Aussagen spezielle erststufige Formeln sind – so genannte *tgds* (engl. *tuple-generating dependencies*) und *egds* (engl. *equality-generating dependencies*), zu deren Definition an dieser Stelle auf Abschnitt 2 verwiesen sei. Wir werden uns hier der Einfachheit halber auf solche Schema-Mappings beschränken.

Üblicherweise existieren zu einer Quelldatenbank S mehrere Zieldatenbanken T, in die S gemäß eines Schema-Mappings M übersetzt werden kann. Solche Zieldatenbanken T heißen *Lösungen* für S unter M. Das wirft sofort zwei grundlegende Fragen auf. Die Erste

ist, welche Lösung zu einer gegebenen Quelldatenbank berechnet werden soll. In vielen Fällen haben sich hier die in [FKMP05] eingeführten *universellen Lösungen* bewährt.

Die zweite Frage ist, wie Anfragen über dem Zielschema (d.h. Anfragen an das Resultat des Datenaustauschs) beantwortet werden können, so dass die Antworten semantisch konsistent mit der Quelldatenbank sind. Das Problem besteht hier in der Existenz unterschiedlicher Lösungen. Auf welche dieser Lösungen sollte man sich zur Beantwortung der Anfrage also beziehen? Diese Frage ist nicht nur für den relationalen Datenaustausch wichtig, sondern auch in der Informationsintegration. Eine Antwort auf diese Frage liefert insbesondere ein wichtiges Kriterium zur Auswahl der zu berechnenden Lösung. Es herrscht weitgehend Einigkeit darüber, dass zur Beantwortung so genannter *Vereinigungen konjunktiver Anfragen* (einer Klasse von Anfragen, die viele der in der Praxis gestellten Anfragen abdeckt) die in [FKMP05] vorgestellte *Sichere Antworten-Semantik* gut geeignet ist. Unter dieser Semantik werden Anfragen durch die Menge aller Tupel beantwortet, die unabhängig von der konkreten Lösung eine Antwort zur Anfrage sind. Um die sicheren Antworten für solche Anfragen zu bestimmen, reicht es im Prinzip aus, universelle Lösungen zu berechnen und die entsprechende Anfrage darauf auszuwerten [FKMP05]. Ähnliche Resultate gelten auch für andere monotone Anfragen [DNR08]. Für viele nichtmonotone Anfragen liefert die Sichere Antworten-Semantik allerdings unintuitive Antworten. Die Frage nach einer geeigneten Semantik für nicht-monotone Anfragen und wie man nicht-monotone Anfragen unter einer solchen Semantik auswertet hat sich als weitaus schwieriger herausgestellt.

Diese Arbeit fasst die Hauptergebnisse meiner Dissertation [Her10] zusammen. Sie beschäftigt sich zum Einen mit der Berechnung der sicheren Antworten für monotone Anfragen und zum Anderen mit Semantiken für nicht-monotone Anfragen und der Komplexität der Auswertung solcher Anfragen unter diesen Semantiken. Die Hauptresultate lassen sich in drei Bereiche einteilen:

1. *Die Komplexität der Berechnung universeller Lösungen.* Wie bereits angesprochen sind universelle Lösungen zur Berechnung der sicheren Antworten zu Vereinigungen konjunktiver Anfragen von Bedeutung. Wir konstruieren hier ein Schema-Mapping M, bei dem unentscheidbar ist, ob eine gegebene Quelldatenbank eine universelle Lösung unter M besitzt. Bei der Konstruktion von Schema-Mappings muss man also sorgfältig sein, wenn wichtige Aufgaben durchführbar sein sollen. Bemerkenswert ist hierbei, dass M fest und nicht Teil der Eingabe ist. Außerdem verwendet M nur tgds. Der Beweis verstärkt insbesondere Resultate aus [DNR08].

2. *Semantiken für nicht-monotone Anfragen.* Hier geht es um Semantiken für die Beantwortung nicht-monotoner Anfragen. Zum Einen wird die von Libkin in [Lib06] entwickelte CWA-Semantik auf eine größere Klasse von Schema-Mappings erweitert und untersucht. Die CWA-Semantik war die erste Semantik speziell für nichtmonotone Anfragen und liefert in vielen Fällen, in denen die sicheren Antworten unintuitiv sind, die gewünschten Resultate. Zum Anderen wird die GCWA*-Semantik eingeführt, die einen etwas anderen Ansatz verfolgt und auf Ideen aus dem Bereich der deduktiven Datenbanken basiert. Einer der Vorteile dieser Semantik ist, dass sie im Unterschied zur CWA-Semantik nicht syntaktisch, sondern semantisch orientiert ist, d.h. Anfragen an logisch äquivalente Schema-Mappings gleich beantwortet.

3. *Die Komplexität der Beantwortung nicht-monotoner Anfragen.* Hier wird die Komplexität der Beantwortung nicht-monotoner Anfragen unter den in 2. angesprochenen Semantiken untersucht. Wir konzentrieren uns dabei auf die *Datenkomplexität* (die Komplexität bei festem Schema-Mapping und fester Anfrage). Die Auswertung nicht-monotoner Anfragen ist in einigen Fällen sehr schwierig: co-NP- bzw. NP-schwer bzw. sogar unentscheidbar. Das Hauptresultat und der am technischsten herausforderndste Teil ist ein Beweis, dass sich die so genannten *universellen Anfragen* unter hinreichend, aber nicht allzu stark eingeschränkten Schema-Mappings unter der GCWA*-Semantik in Polynomialzeit auswerten lassen.

Der restliche Teil dieser Arbeit ist wie folgt strukturiert. In Abschnitt 2 werden grundlegende Begriffe eingeführt, auf die wir in den nachfolgenden Abschnitten immer wieder zurückgreifen. Abschnitt 3 bespricht die Sichere Antworten-Semantik und Anfrageauswertung unter dieser Semantik. Unter Anderem setzen wir uns hier mit der Komplexität der Berechnung universeller Lösungen auseinander. Abschnitt 4 stellt Semantiken zur Beantwortung nicht-monotoner Anfragen vor und Abschnitt 5 beschäftigt sich mit der Komplexität der Auswertung nicht-monotoner Anfragen unter diesen Semantiken.

Für eine ausführliche Darstellung der hier vorgestellten Ergebnisse sei auf meine Dissertation [Her10] verwiesen.

2 Grundbegriffe aus der Datenbanktheorie

Dieser Abschnitt gibt einen kurzen Überblick über die wichtigsten Grundbegriffe aus der Datenbanktheorie. Eine umfassende Einführung gibt [AHV95].

Ein *Schema* σ ist eine endliche Menge von Relationssymbolen, wobei jedes Relationssymbol R eine Stelligkeit $\mathrm{ar}(R)$ hat. Eine σ-*Instanz* I ordnet jedem $R \in \sigma$ eine endliche Relation R^I der Stelligkeit $\mathrm{ar}(R)$ zu. Mit $\mathrm{dom}(I)$ bezeichnen wir die Menge aller in I vorkommenden Werte, d.h. derjenigen Elemente v, für die ein $R \in \sigma$, ein Tupel $(v_1, \ldots, v_{\mathrm{ar}(R)}) \in R^I$ und ein $i \in \{1, \ldots, \mathrm{ar}(R)\}$ existiert, so dass $v = v_i$. Wir nehmen an, dass $\mathrm{dom}(I) \subseteq \mathit{Dom}$ für eine feste, unendliche Menge *Dom*. Außerdem sei *Dom* die Vereinigung zweier disjunkter, unendlicher Mengen *Const* und *Null*, deren Elemente wir *Konstanten* bzw. *Nulls* nennen. Nulls dienen als Platzhalter für Konstanten, können also als Variablen angesehen werden. Ein *Atom* ist ein Ausdruck der Form $R(v_1, \ldots, v_{\mathrm{ar}(r)})$, so dass $R \in \sigma$ und jedes v_i in *Dom* liegt. Wir schreiben $R(\bar{v}) \in I$ an Stelle von $\bar{v} \in R^I$ und identifizieren Instanzen mit der Menge der Atome $R(\bar{v})$ mit $R(\bar{v}) \in I$. Für eine σ-Instanz I und eine τ-Instanz J bedeutet dann $I \subseteq J$, dass alle Atome aus I auch in J vorkommen, und $I \cup J$ bezeichnet die $\sigma \cup \tau$-Instanz, die aus allen Atomen aus I und J besteht.

Beispiel 1. Sei $\tau = \{\mathit{Autoren}, \mathit{BücherInfos}\}$ mit zweistelligen Relationssymbolen *Autoren* und *BücherInfos*. Die Datenbank

Autoren:

Name	Buch_ID
S. Arora	1
B. Barak	1
W. Hodges	2

BücherInfos:

ID	Titel
1	Computational Complexity
2	Model Theory

können wir dann durch folgende τ-Instanz repräsentieren:

$$T = \{\mathit{Autoren}(\text{S. Arora}, 1), \mathit{Autoren}(\text{B. Barak}, 1), \mathit{Autoren}(\text{W. Hodges}, 2),$$
$$\mathit{BücherInfos}(1, \text{Computational Complexity}), \mathit{BücherInfos}(2, \text{Model Theory})\}.$$

Wir nehmen an, dass alle in der Datenbank vorkommenden Werte aus *Const* stammen. □

Erststufige Anfragen. Anfragen an Instanzen und Aussagen in Schema-Mappings formulieren wir in erststufiger Logik (FO). Atomare FO-Formeln über einem Schema σ sind Formeln der Gestalt $R(u_1, \ldots, u_{\text{ar}(R)})$ oder $u_1 = u_2$, wobei $R \in \sigma$ und jedes u_i eine Variable oder ein Element aus *Const* ist. Formeln der ersten Gestalt nennen wir *Relationsatome* (über σ). Wenn φ und ψ FO-Formeln über σ sind und x eine Variable ist, dann sind auch $\neg\varphi$, $(\varphi \star \psi)$ für $\star \in \{\wedge, \vee, \rightarrow\}$, $\exists x\,\varphi$ und $\forall x\,\varphi$ FO-Formeln über σ. Die Menge $\text{frei}(\varphi)$ der in φ frei vorkommenden Variablen ist wie üblich definiert. Für eine FO-Formel φ und ein Tupel $\bar{x} = (x_1, \ldots, x_k)$ von Variablen deutet die Schreibweise $\varphi(\bar{x})$ an, dass $\text{frei}(\varphi) = \{x_1, \ldots, x_k\}$.

Eine Belegung für eine FO-Formel φ über σ in einer σ-Instanz I ist eine Abbildung $\alpha\colon \text{frei}(\varphi) \to \text{dom}(I) \cup \text{const}(\varphi)$ (mit $\text{const}(\varphi)$ bezeichnen wir die Menge der Konstanten in φ), die wir so auf *Const* erweitern, dass für alle $c \in \mathit{Const}$ gilt: $\alpha(c) = c$. Wir schreiben $(I, \alpha) \models \varphi$, um anzudeuten, dass φ in I unter der Belegung α erfüllt ist. Die Relation $\models$ ist wie üblich definiert. Beispielsweise gilt $(I, \alpha) \models R(u_1, \ldots, u_{\text{ar}(R)})$ genau dann, wenn $(\alpha(u_1), \ldots, \alpha(u_{\text{ar}(R)})) \in R^I$; $(I, \alpha) \models u_1 = u_2$ genau dann, wenn $\alpha(u_1) = \alpha(u_2)$; und $(I, \alpha) \models \exists x\,\varphi$ genau dann, wenn ein $v \in \text{dom}(I) \cup \text{const}(\varphi)$ existiert mit $(I, \alpha[v/x]) \models \varphi$, wobei die Belegung $\alpha[v/x]$ wie α definiert ist, nur dass x der Wert v zugeordnet wird.

Eine *FO-Anfrage* über σ besteht aus einer FO-Formel φ über σ und einem Tupel $\bar{x} = (x_1, \ldots, x_k)$ der freien Variablen von φ; wir notieren solche Anfragen als $\varphi(\bar{x})$. Für eine σ-Instanz I und ein Tupel $\bar{v} = (v_1, \ldots, v_k)$ über *Dom* schreiben wir $I \models \varphi[\bar{v}]$, falls $(I, \alpha) \models \varphi$ für die Belegung α mit $\alpha(x_i) = v_i$. Das *Resultat* von $\varphi(\bar{x})$ auf I, geschrieben $[\![\varphi(\bar{x})]\!]^I$, ist die Menge aller Tupel $\bar{v} \in \mathit{Dom}^k$ mit $I \models \varphi[\bar{v}]$.

Eine *Vereinigung konjunktiver Anfragen (UCQ)* ist eine FO-Anfrage, die nur aus atomaren FO-Formeln, Konjunktion, Disjunktion und Existenzquantoren aufgebaut ist.

Beispiel 2. Seien τ und T wie in Beispiel 1. Die folgende FO-Anfrage über τ ist dann ein UCQ: $\varphi(x) := \exists y\big(\mathit{BücherInfos}(y, \text{Computational Complexity}) \wedge \mathit{Autoren}(x, y)\big)$. Das Resultat von $\varphi(x)$ auf T ist die Menge der Autoren von *Computational Complexity*, d.h. $[\![\varphi(x)]\!]^T = \{\text{S. Arora}, \text{B. Barak}\}$. □

Schema-Mappings. Ein *Schema-Mapping* $M = (\sigma, \tau, \Sigma)$ besteht aus zwei disjunkten Schemata σ, τ, dem *Quell-* und dem *Zielschema*, sowie einer endlichen Menge Σ von Aussagen, bei denen wir zwischen *st-tgds*, *t-tgds* und *egds* unterscheiden. *St-tgds* (engl. *source-to-target tuple-generating dependencies*) sind FO-Formeln der Gestalt $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x}, \bar{z}))$, wobei φ eine Konjunktion von Relationsatomen über σ ist, ψ eine Konjunktion von Relationsatomen über τ ist und in φ, ψ keine Konstanten vorkommen. Ähnlich sind *t-tgds* (engl. *target tuple-generating dependencies*) definiert. Der einzige Unterschied besteht darin, dass φ wie ψ eine Konjunktion von Relationsatomen über τ ist. *Egds* (engl.

equality-generating dependencies) sind FO-Formeln der Gestalt $\forall \bar{x}(\varphi(\bar{x}) \rightarrow y = z)$, wobei φ eine Konjunktion von Relationsatomen über τ ist, y und z in $\bar{x}$ vorkommen und φ keine Konstanten enthält.

Eine *Quellinstanz* S für M ist eine σ-Instanz, die *keine* Nulls enthält. Eine *Lösung für S unter M* ist eine τ-Instanz T, so dass $S \cup T$ alle Aussagen in Σ erfüllt.

Beispiel 3. Sei $\sigma = \{\mathit{Bücher}\}$ mit einem zweistelligen Relationssymbol *Bücher* und sei τ wie in Beispiel 1. Angenommen, Quelldatenbanken vom Schema σ speichern in der Relation *Bücher* Tupel der Form (Buchtitel, Autor), wie z.B. in der σ-Instanz

$$S = \{\mathit{Bücher}(\text{Comput. Compl.}, \text{S. Arora}), \mathit{Bücher}(\text{Comput. Compl.}, \text{B. Barak}), \\ \mathit{Bücher}(\text{Model Theory}, \text{W. Hodges})\}.$$

Solche σ-Instanzen könnten dann in τ-Instanzen übersetzt werden, z.B. mit Hilfe des Schema-Mappings $M = (\sigma, \tau, \Sigma)$, bei dem Σ das st-tgd und das egd

$$\forall x \forall y \big(\mathit{Bücher}(x,y) \rightarrow \exists z(\mathit{Autoren}(y,z) \wedge \mathit{BücherInfos}(z,x))\big), \\ \forall x_1 \forall x_2 \forall y \big(\mathit{BücherInfos}(x_1,y) \wedge \mathit{BücherInfos}(x_2,y) \rightarrow x_1 = x_2\big)$$

enthält. Hier ist S eine Quellinstanz für M und die Instanz T aus Beispiel 1 eine Lösung für S unter M. Andere Lösungen für S ergeben sich, wenn man in T die Einträge 1 und 2 durch verschiedene Werte v_1, v_2 ersetzt oder neue Tupel (v_1, v_2) zu $\mathit{Autoren}^T$ hinzufügt. □

3 Sichere Antworten und universelle Lösungen

Anfrageverarbeitung im relationalen Datenaustausch heißt, Anfragen über dem Zielschema mit den Informationen der Quellinstanz und des Schema-Mappings zu beantworten. Die Anfrage $q(x)$ aus Beispiel 2 ist eine mögliche Anfrage über dem Zielschema des Schema-Mappings aus Beispiel 3; hier ist intuitiv klar, dass die Antwort die Menge $[\![q(x)]\!]^T$ aus Beispiel 2 sein sollte. Eine häufige Annahme beim relationalen Datenaustausch ist, dass die Quellinstanz nach der Übersetzung nicht mehr zur Verfügung steht und die Anfrage mit Hilfe einer Lösung beantwortet werden muss. Da Quellinstanzen in der Regel mehr als eine Lösung besitzen, muss man sich fragen, auf welche Lösungen man sich zur Beantwortung der Anfrage beziehen soll. Fagin, Kolaitis, Miller und Popa führen in [FKMP05] die *Sichere Antworten-Semantik* ein und zeigen, dass viele interessante Anfragen unter dieser Semantik mit universellen Lösungen ausgewertet werden können.

Definition 4 (Sichere Antworten [FKMP05])**.** Sei M ein Schema-Mapping, S eine Quellinstanz für M und $q(\bar{x})$ eine Anfrage über dem Zielschema von M. Die Menge der *sicheren Antworten* auf $q(\bar{x})$ unter M und S besteht aus allen Tupeln $\bar{a} \in \mathit{Dom}^{|\bar{x}|}$, so dass für alle Lösungen T für S unter M gilt: $\bar{a} \in [\![q(\bar{x})]\!]^T$.

Die sicheren Antworten einer Anfrage sind also all jene Antworten, die immer im Anfrageresultat enthalten sind, egal auf welcher Lösung die Anfrage beantwortet wird.

Eine *universelle Lösung* für eine Quellinstanz S unter einem Schema-Mapping M ist eine Lösung T, so dass für *alle* Lösungen T' für S unter M ein Homomorphismus von T nach

T' existiert [FKMP05]. Hierbei ist ein Homomorphismus von T nach T' eine struktur- und konstantenbewahrende Abbildung $h\colon \mathrm{dom}(T) \to \mathrm{dom}(T')$, d.h. $h(c) = c$ für alle Konstanten $c \in \mathrm{dom}(T)$ und für alle $R(v_1,\ldots,v_k) \in T$ gilt: $R(h(v_1),\ldots,h(v_k)) \in T'$. Gewissermaßen sind universelle Lösungen allgemeinste Lösungen. Wenn wir zu Beispiel 3 zurückkehren, so ist die Instanz, die man aus T durch Ersetzen der Konstanten 1 und 2 durch verschiedene Nulls $\perp_1$ und $\perp_2$ erhält, eine universelle Lösung für S unter M.

Wie in [FKMP05] gezeigt, gilt für alle Schema-Mappings M, alle Quellinstanzen S, alle universellen Lösungen T für S unter M und alle UCQs $q(\bar{x})$ über dem Zielschema von M, dass die sicheren Antworten auf $q(\bar{x})$ unter M und S gerade die Tupel in $[\![q(\bar{x})]\!]^T$ sind, die keine Nulls enthalten. Die Berechnung der sicheren Antworten auf UCQs reduziert sich also im Wesentlichen auf die Berechnung universeller Lösungen. Leider existieren universelle Lösungen nicht in jedem Fall. In einer mit [FKMP05] beginnenden Serie von Arbeiten wurden immer schwächere Einschränkungen an Schema-Mappings präsentiert, die es für feste Schema-Mappings ermöglichen, in Polynomialzeit festzustellen, ob eine Quellinstanz eine universelle Lösung besitzt, und – wenn ja – eine zu berechnen. Die Entscheidbarkeit der Existenz universeller Lösungen blieb aber offen und wurde letztendlich in einem recht starken Sinn negativ beantwortet:

Satz 5 ([Her10])**.** *Es gibt ein Schema-Mapping $M = (\sigma, \tau, \Sigma)$, bei dem Σ nur aus st-tgds und t-tgds besteht, so dass folgendes Problem unentscheidbar ist: Gegeben eine Quellinstanz S für M, gibt es eine universelle Lösung für S unter M?*

Aus dem Beweis folgt auch, dass das Problem, ob die zur Berechnung universeller Lösungen eingesetzte *Chase-Prozedur* für die Menge Σ in M bei gegebener Quellinstanz terminiert, unentscheidbar ist. Die oben genannten Resultate verstärken insbesondere Ergebnisse aus [DNR08].

4 Semantiken für nicht-monotone Anfragen

Man hat schnell realisiert, dass die Sichere Antworten-Semantik für nicht-monotone Anfragen nicht immer die gewünschten Resultate liefert. Dazu folgendes Beispiel:

Beispiel 6. Wir betrachten das *Kopier-Schema-Mapping* $M = (\{R\}, \{R'\}, \Sigma)$, bei dem R, R' zweistellige Relationssymbole sind und Σ nur das st-tgd $\forall x \forall y\big(R(x,y) \to R'(x,y)\big)$ enthält. Führt man sich vor Augen, dass ein Schema-Mapping eine Transformation von Quell- in Zieldaten beschreibt, so ist es natürlich, anzunehmen, dass die Relation R' nach der Transformation eine *Kopie* von R ist. Insbesondere sollte das Resultat einer Anfrage $q(\bar{x})$ über $\{R'\}$ sich nicht vom Resultat der entsprechenden Anfrage $q'(\bar{x})$ über $\{R\}$, bei der R' nur durch R ausgetauscht ist, unterscheiden. Bei der FO-Anfrage $q(x) := \exists y\big(R'(x,y) \wedge \neg R'(y,x)\big)$ und der Quellinstanz $S = \{R(a,b)\}$ würde man also das Resultat $\{a\}$ erwarten. Die sicheren Antworten auf $q(x)$ unter M und S sind jedoch leer. □

In der Literatur existieren verschiedene Ansätze für Semantiken, unter denen nicht-monotone Anfragen sinnvoll beantwortet werden können. Wir besprechen im Folgenden die *CWA-Semantik* und die *GCWA*-Semantik*. Neben diesen Semantiken gibt es noch zwei weitere Semantiken, die 2008 von Libkin und Sirangelo sowie Afrati und Kolaitis untersucht wurden und im Wesentlichen Weiterentwicklungen der CWA-Semantik darstellen.

4.1 Die CWA-Semantik

Die CWA-Semantik war die erste Semantik, die speziell zur Beantwortung nicht-monotoner Anfragen entwickelt wurde. Libkin [Lib06] konzipierte die CWA-Semantik für Schema-Mappings mit st-tgds und [Her10] erweiterte sie auf den generelleren Fall von Schema-Mappings mit st-tgds, t-tgds und egds. Wie auch die späteren Semantiken basiert diese auf der Idee „unerwünschte“ Daten aus Lösungen “auszublenden“ bzw. die sicheren Antworten nur bzgl. der Menge der Lösungen zu berechnen, die solche Daten nicht beinhalten (in Beispiel 6 würden wir nur Kopien der Quellinstanz als Lösung zulassen). Prinzipiell unterscheiden sich die verschiedenen Semantiken nur darin, nach welcher Regel Daten ausgeblendet werden. Alle Semantiken basieren auf einer Variante der so genannten *Closed World Assumption (CWA)*, die intuitiv alle Daten als unzulässig ansieht, die nicht aus den Quelldaten und dem Schema-Mapping folgen.

Bei der CWA-Semantik formalisiert man die zulässigen Lösungen durch *CWA-Lösungen*. CWA-Lösungen basieren im folgenden Sinn auf der CWA:

1. Alle Atome der CWA-Lösung müssen durch das Schema-Mapping und die Quellinstanz in einem bestimmten Sinn *gerechtfertigt* sein.
2. Jede mögliche Rechtfertigung darf nur einmal verwendet werden.
3. Die CWA-Lösung enthält nur *Fakten* (Aussagen, die sich durch konjunktive Anfragen ohne freie Variablen ausdrücken lassen), die aus dem Schema-Mapping und der Quellinstanz logisch folgen.

Bei der Erweiterung auf Schema-Mappings mit tgds und egds in [Her10] besteht die Hauptschwierigkeit in der Formalisierung der Anforderungen 1 und 2. Dies wird in [Her10] durch einen ableitungsbasierten Ansatz mittels einer passend kontrollierten Variante der *Chase-Prozedur* realisiert. Zudem wird dort ein *2-Personen-Spiel* eingeführt, mit dem sich diese Anforderungen charakterisieren lassen. Es zeigt sich, dass CWA-Lösungen spezielle *universelle Lösungen* sind und eine Quellinstanz genau dann eine CWA-Lösung besitzt, wenn sie eine universelle Lösung besitzt. Aus Satz 5 folgt also unmittelbar die Unentscheidbarkeit der Existenz von CWA-Lösungen. Außerdem ist der von Fagin, Kolaitis und Popa eingeführte *Kern der universellen Lösungen* die „kleinste“ CWA-Lösung (bis auf Umbenennung von Nulls).

Anfragen werden in der CWA-Semantik dadurch beantwortet, dass man die sicheren Antworten bzgl. CWA-Lösungen berechnet[1]; allerdings wird zur Beantwortung von Anfragen auf individuellen CWA-Lösungen eine Semantik verwendet, die speziell für die Beantwortung von Anfragen mit Nulls konzipiert wurde. Wir werden hier nicht näher darauf eingehen. In vielen Fällen läßt sich die Auswertung von Anfragen unter der CWA-Semantik – ähnlich wie in Abschnitt 3 für die sicheren Antworten beschrieben – auf die Auswertung auf einzelnen CWA-Lösungen zurückzuführen.

4.2 Die GCWA*-Semantik

Die CWA-Semantik liefert in vielen Fällen, in denen die sicheren Antworten unintuitiv sind, die gewünschten Resultate. So zum Beispiel für die Anfrage in Beispiel 6. Zwei

[1] Genau genommen ergibt dies nur eine der vier in [Lib06, Her10] eingeführten CWA-Semantiken. Der Einfachheit halber gehen wir hier aber nicht näher darauf ein.

Eigenschaften, die recht natürlich scheinen, fehlen ihr aber: Erstens ist sie nicht invariant unter logisch äquivalenten Schema-Mappings, d.h. Anfragen an die Zielschemata verschiedener Schema-Mappings M_1, M_2 der Form $M_i = (\sigma, \tau, \Sigma_i)$ mit logisch äquivalenten Mengen Σ_1 und Σ_2 können verschieden beantwortet werden. Zweitens reflektiert sie nicht notwendigerweise die gewohnte Semantik von tgds:

Beispiel 7. Sei $M = (\{P\}, \{R\}, \Sigma)$, wobei P einstellig und R zweistellig ist, und Σ aus dem tgd $\theta := \forall x\big(P(x) \rightarrow \exists y R(x,y)\big)$ besteht. Sei $q(x) := \forall x \forall y \forall y' \big(R(x,y) \wedge R(x,y') \rightarrow y = y'\big)$. Unter der CWA-Semantik liefert $q(x)$ für die Quellinstanz $S = \{P(a)\}$ die Antwortmenge $\{a\}$, was intuitiv besagt, dass es genau ein Tupel der Form $(a, \cdot)$ in R gibt. Allerdings sagen S und θ ja wegen $P(a) \in S$ aus, dass *mindestens ein b* mit $R(a,b)$ existiert. D.h. es existiert ein solches b oder zwei oder drei usw. Die CWA-Semantik spiegelt das nicht wieder. □

Die in [Her10] eingeführte GCWA*-Semantik ist invariant unter logisch äquivalenten Schema-Mappings und reflektiert intuitiv die gewohnte Semantik der Constraints. Letzteres wird in [Her10] ausführlich argumentiert. Inspiriert wurde die GCWA*-Semantik durch verschiedene CWA-basierte Semantiken aus dem Bereich der deduktiven Datenbanken, die in [Her10] zuerst in den Kontext des relationalen Datenaustauschs übertragen und dann auf ihre Tauglichkeit zur Beantwortung nicht-monotoner Anfragen in diesem Kontext hin untersucht wurden. Es stellte sich heraus, dass diese Semantiken zu stark oder zu schwach sind bzw. nicht die erforderlichen Eigenschaften aufweisen, bildeten aber dennoch einen guten Ausgangspunkt für die GCWA*-Semantik. Anders als die anderen Semantiken ist die GCWA*-Semantik nicht nur für Schema-Mappings mit st-tgds (und evtl. t-tgds und egds) definiert, sondern generell für alle möglichen Schema-Mappings, die in der Literatur betrachtet werden. Weiterhin ist für Schema-Mappings mit st-tgds und egds die Definition vergleichsweise einfach:

Definition 8 (GCWA*-Semantik [Her10])**.** Sei M ein Schema-Mapping mit st-tgds und egds und sei S eine Quellinstanz für M. Eine *GCWA*-Lösung* für S unter M ist eine Lösung für S unter M ohne Nulls, die eine Vereinigung minimaler Lösungen für S unter M ist. Die Menge der *GCWA*-Antworten* auf eine Anfrage $q(\bar{x})$ über M's Zielschema besteht aus allen Tupeln $\bar{a} \in Dom^{|\bar{x}|}$, so dass für alle GCWA*-Lösungen T für S unter M gilt: $\bar{a} \in [\![q(\bar{x})]\!]^T$.

Für Schema-Mappings wie in Definition 8 lassen sich GCWA*-Lösungen auch analog zur Definition von Lösungen unter der so genannten *generalisierten CWA (GCWA)* für deduktive Datenbanken charakterisieren.

5 Komplexität der Auswertung nicht-monotoner Anfragen

Schließlich wollen wir uns der Frage nach der Komplexität der Beantwortung von Anfragen unter den in Abschnitt 4 vorgestellten Semantiken zuwenden. Wir konzentrieren uns auf die *Datenkomplexität* dieses Problems, d.h. die Komplexität bzgl. eines *festen* Schema-Mappings und einer *festen* Anfrage. Der Vollständigkeit halber sei erwähnt, dass die meisten der in Abschnitt 4 besprochenen Semantiken für monotone Anfragen mit der Sicheren Antworten-Semantik übereinstimmen, d.h. Resultate für die Sichere Antworten-Semantik gehen unmittelbar auf diese Semantiken über.

Generell hat die Auswertung nicht-monotoner Anfragen unter all den angesprochenen Semantiken eine hohe Komplexität. Bei den CWA-Semantiken und der GCWA*-Semantik gibt es z.B. einfache Schema-Mappings und sehr einfache *existenzielle Anfragen* (FO-Anfragen der Form $\exists \bar{x}\, \varphi$, wobei φ quantorenfrei ist), bei denen dieses Problem je nach Semantik co-NP- bzw. NP-schwer ist. Bei der GCWA*-Semantik kann es sogar schon für durch st-tgds definierte Schema-Mappings co-NP-schwer sein. Lässt man in der Anfrage neben Existenzquantoren auch Allquantoren zu, so kann es unentscheidbar werden. Man muss allerdings beachten, dass die Resultate die Datenkomplexität betreffen, d.h. sie zeigen, dass es ein Schema-Mapping M und eine Anfrage $q(\bar{x})$ gibt, für die das entsprechende Problem schwer ist. Das bedeutet nicht, dass es für alle solchen Schema-Mappings und Anfragen schwer ist.

In Anbetracht des gerade erwähnten Unentscheidbarkeitsresultats mag das folgende Resultat überraschend sein. Es betrifft die Klasse der *universellen Anfragen*, d.h. FO-Anfragen der Gestalt $\forall \bar{x}\, \varphi$, wobei φ quantorenfrei ist, und besagt, dass das Auswertungsproblem für durch st-tgds definierte Schema-Mappings und *alle* universellen Anfragen unter der GCWA*-Semantik in co-NP liegt. Außerdem wird gezeigt, dass es für Schema-Mappings M mit *gepackten* st-tgds und universelle Anfragen $q(\bar{x})$ in Polynomialzeit möglich ist, bei Eingabe des Kerns der universellen Lösungen (im Folgenden kurz *Kernlösung*) für eine Quellinstanz S unter M die GCWA*-Antworten auf $q(\bar{x})$ unter M und S zu berechnen. Ein st-tgd der Gestalt $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$ ist dabei *gepackt*, wenn je zwei verschiedene Relationsatome in ψ eine Variable aus $\bar{z}$ gemeinsam haben. Man zeigt leicht, dass jedes st-tgd $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$, das höchstens zwei Relationsatome in ψ mit Variablen aus $\bar{z}$ besitzt, logisch äquivalent zu einer Menge von gepackten st-tgds ist, die höchstens so groß wie die Anzahl der Relationsatome in ψ ist. Durch gepackte st-tgds spezifizierte Schema-Mappings bilden somit eine interessante Klasse von Schema-Mappings. Wir können nun das Hauptresultat formulieren:

Satz 9 ([Her10])**.** *Sei M ein Schema-Mapping mit gepackten st-tgds und sei $q(\bar{x})$ eine universelle Anfrage über M's Zielschema. Dann gibt es einen Polynomialzeitalgorithmus, der bei Eingabe der Kernlösung für eine Quellinstanz S für M die Menge der GCWA*-Antworten auf $q(\bar{x})$ unter M und S ausgibt.*

Es ist bekannt, dass für Schema-Mappings M mit st-tgds die Kernlösungen für eine Quellinstanz effizient berechnet werden kann (wenn M fest ist). Somit liefert obiger Algorithmus auch einen Polynomialzeitalgorithmus zur Berechnung der GCWA*-Antworten bei Eingabe einer Quellinstanz für M. Interessant ist hier, dass die Kernlösung auch zur Auswertung von UCQs und anderen Anfragen unter der Sicheren Antworten-Semantik verwendet werden kann. Man muss somit nur diese eine Lösung vorhalten, um solche Anfragen und auch universelle Anfragen zu beantworten. Es ist ebenfalls interessant, dass das Schema-Mapping, das im oben angesprochenen Unentscheidbarkeitsbeweis konstruiert wird, aus gepackten st-tgds besteht.

Theorem 9 ist das technisch herausforderndste Resultat in [Her10]. Im Wesentlichen reduziert man das Problem zuerst auf ein *Erfüllbarkeitsproblem* für *existentielle Anfragen* über einer bestimmten Menge von Lösungen: der Menge der Lösungen, die sich als Vereinigung von inklusionsminimalen Bildern von Homomorphismen auf der Kernlösung dar-

stellen lassen. Ein wichtiges Teilproblem, das hierzu gelöst wird, ist, für ein gegebenes Atom ein inklusionsminimales homomorphes Bild der Kernlösung zu finden, das dieses Atom enthält. Im Allgemeinen existieren unendlich viele solcher Bilder; es reicht aber aus, endlich viele Repräsentanten zu betrachten. Der Hauptteil des Beweises besteht darin, aus den im Allgemeinen exponentiell vielen Repräsentanten polynomiell viele herauszufinden, auf die man sich bei der Suche nach einem geeigneten Repräsentanten beschränken kann. Hierzu nutzen wir unter Anderem die besondere Struktur universeller Lösungen unter Schema-Mappings mit st-tgds aus, die bereits in anderen Arbeiten zum Thema relationaler Datenaustausch eine wichtige Rolle spielte.

Literatur

[ABLM10] Marcelo Arenas, Pablo Barceló, Leonid Libkin und Filip Murlak. *Relational and XML Data Exchange*. Morgan & Claypool, 2010.

[AHV95] Serge Abiteboul, Richard Hull und Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Bar09] Pablo Barceló. Logical Foundations of Relational Data Exchange. *SIGMOD Record*, 38(1):49–58, 2009.

[DNR08] Alin Deutsch, Alan Nash und Jeff Remmel. The Chase Revisited. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS)*, Seiten 149–158, 2008.

[FKMP05] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller und Lucian Popa. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[Her10] André Hernich. *Foundations of Query Answering in Relational Data Exchange*. Dissertation, Institut für Informatik, Goethe-Universität Frankfurt am Main, 2010. Veröffentlicht beim Logos Verlag Berlin, ISBN 978-3-8325-2735-8, 2010.

[Kol05] Phokion G. Kolaitis. Schema Mappings, Data Exchange, and Metadata Management. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS)*, Seiten 61–75, 2005.

[Lib06] Leonid Libkin. Data Exchange and Incomplete Information. In *Proceedings of the 25th ACM Symposium on Principles of Database Systems (PODS)*, Seiten 60–69, 2006.

André Hernich wurde am 22. Januar 1980 in Prenzlau geboren. Zum Wintersemester 2000/01 begann er ein Informatikstudium an der Technischen Universität Berlin, das er im Februar 2005 mit einem Diplom abschloss. Seine Diplomarbeit schrieb er am Lehrstuhl *Theoretische Informatik – Algorithmik und Logik* von Prof. Dr. Siefkes. Nach dem Studium war er als wissenschaftlicher Mitarbeiter tätig und arbeitete an seiner Promotion, zuerst an der Humboldt-Universität zu Berlin und dann an der Goethe-Universität Frankfurt am Main, jeweils in der Arbeitsgruppe von Prof. Dr. Schweikardt. Seit April 2010 ist er wissenschaftlicher Mitarbeiter in der Arbeitsgruppe *Logik in der Informatik* von Prof. Dr. Grohe an der Humboldt-Universität zu Berlin.