INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

Gesicherte und mögliche Antworten auf Anfragen an relationale Datenbanken mit partiellen Relationen

Hans-Joachim Klein

Bericht Nr. 9802
Dezember 1997

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik und Praktische Mathematik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Gesicherte und mögliche Antworten auf Anfragen an relationale Datenbanken mit partiellen Relationen

Hans-Joachim Klein

Bericht Nr. 9802
Dezember 1997

e-mail: hjk@informatik.uni-kiel.de

Dieser Bericht gibt den Inhalt der Habilitationsschrift wieder, die der Verfasser im Januar 1997 bei der Technischen Fakultät der Christian–Albrechts–Universität zu Kiel eingereicht hat.
Datum der Habilitation: 2. Juli 1997

Abstract

In the book "The Theory of Relational Databases" written by David Maier in 1983, the following warning can be found in the introduction to a chapter on incomplete information and database semantics: *The reader is warned that the topics in this chapter are matters of personal taste. Whether the definitions and approaches offered seem right depends on individual intuition.*

Although the literature of incomplete information in relational databases is numerous, no approach in this field proposed since the time of that writing (and before) has been considered a final solution. In particular there are many objections to constructs offered by the de facto standard query language SQL for handling incomplete information.

In this thesis we deal with the problem in a way different from most previous approaches. We take a denotational approach insofar as we start with definitions of different kinds of answers to queries by considering queries as functions. The meaning of answers is determined without references to definitions of operations or to the interpretation of logical query expressions in partial database states. Algorithms for the evaluation of queries formulated in concrete query languages are given later according to these definitions.

We restrict ourselves to two possible meanings of missing values: *a value exists but it is unknown* and *there is no information on a value.* After some introductory definitions we discuss several kinds of answers representing sure and maybe information for the first meaning. Information in answers is sure if it is valid independent of unknown values. Maybe information is valid for at least one possible replacement of unknown values by defined values. Two forms for representing missing values are considered: ω-relations where a single special value is on hand for denoting missing values and V-relations where a set of variables is available for this purpose. For both forms we consider queries formulated in relational algebra as well as in tuple relational calculus. Following the discussion of the relevant literature, several evaluation methods for different kinds of answers are presented and an undecidability result is shown for ω-relations. In case of incomplete information, completeness of all interesting kinds of answers is tied up with the tautology problem. In order not to lose too much information, multi-valued logics are used by some of our efficient evaluation methods. The classical equivalence theorem concerning relational algebra and relational calculus is proven for some extensions to relational algebra operations and modified interpretations of calculus expressions in case of partial relations.

In the last two chapters we concentrate on the second meaning of missing values mentioned above. We show the problems connected with the definition of sure answers under this "no information" assumption and present a transformation method which allows to "adorn" SQL-queries in such a way that answers always represent sure information.

Keywords

relational databases, incomplete information, partial relations, null values, query languages, sure answers, maybe answers, relational algebra, relational calculus, multi-valued logic, SQL semantics

Danksagung

An dieser Stelle möchte ich allen danken, die durch nützliche Hinweise und kritische Bemerkungen zum Gelingen dieser Arbeit beigetragen haben. Hervorheben möchte ich meinen früheren Kollegen Kai Goetzke, der die Anfänge meiner Beschäftigung mit Nullwerten begleitete und stets zu intensiven Diskussionen bereit war. Mein besonderer Dank gilt auch Jochen Rasch für das sorgfältige Lesen von Vorversionen der Arbeit. Ich möchte nicht versäumen, Brigitte, Bennet und Geelke zu nennen, die in den letzten Jahren viel Geduld mit mir haben mußten.

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit beschäftigt sich mit Problemen, die im relationalen Datenmodell beim Übergang von total definierten zu partiell definierten Relationen auftreten. Dabei stehen Fragen im Zusammenhang mit der Semantik von Anfragesprachen und Methoden für die Berechnung verschiedener Formen von Antworten auf Anfragen im Mittelpunkt.

Wissen über einen Sachverhalt wird in einer relationalen Datenbank in Form von Tupeln gespeichert, die eindeutig benannten Mengen, den Relationen, zugeordnet sind. Alle Tupel einer Relation haben den gleichen Typ, der durch die Angabe einer Menge von sogenannten Attributen festgelegt ist. Den Attributen sind eindeutig Wertebereiche zugeordnet, aus denen jeweils genau ein Wert für jedes Attribut bei der Tupelbildung entnommen werden darf. Da Ordnungen innerhalb der Attributmenge einer Relation keine Bedeutung tragen, kann jedes Tupel als Funktion aufgefaßt werden, die Attribute in Werte aus den zugehörigen Wertebereichen abbildet. Mit dieser Sichtweise ist jede Relation eine Menge von Funktionen.

Neben Tupelmengen werden in realen Systemen auch Multimengen von Tupeln zugelassen, die Tabellen genannt werden. Da die von uns diskutierten Probleme für Tabellen keine wesentlich andere Qualität besitzen, beschränken wir uns auf die Betrachtung von Relationen.

In vielen Datenbankanwendungen kann Tupeln nicht immer in jedem Attribut ein Wert des entsprechenden Wertebereichs zugeordnet werden. Die Gründe hierfür können sehr unterschiedlich sein. Ein Wert kann zum Beispiel nicht angebbar sein, weil für den zu modellierenden Sachverhalt eine entsprechende Eigenschaft nicht vorhanden ist (das Lieferdatum für einen nicht mehr lieferbaren Artikel). Häufig ist ein Wert beim Einspeichern eines Tupels auch unbekannt, weil Angaben nicht vollständig gemacht wurden (es fehlt das Geburtsdatum einer Person). Bei der Datenmodellierung, d.h. beim Festlegen von Attributen, Wertebereichen und Relationstypen, kann durch entsprechendes Vorgehen erreicht werden, daß Relationen stets nur total definierte Tupel enthalten (trenne nicht lieferbare Artikel von den anderen Artikeln; füge dem Wertebereich "Datum" einen speziellen Wert hinzu, der für eine fehlende Datumsangabe steht). Solche Lösungen führen aber häufig zu unübersichtlichen, ineffizienten Strukturen und zu inhomogenen Wertebereichen, was beides für die Anwendungsprogrammierung Nachteile bringt. Deshalb sind in realen Datenbanksystemen partiell definierte Relationen bzw. Tabellen zugelassen.

Probleme, die durch undefinierte Werte für Datenbanksysteme entstehen können, wurden schon frühzeitig erkannt, so daß sich auch ein Standardisierungskomitee mit ihnen beschäftigte und eine ganze Reihe von Ursachen für das Fehlen von Werten auflistete ([ANSI75]).

In älteren Datenbanksystemen, etwa CODASYL-Systemen, haben fehlende Werte einen etwas anderen Stellenwert als in relationalen Systemen. Die explizite Repräsentation von Beziehungen zwischen Sätzen durch Mengen von Satzpaaren, den Sets, führt im Fall fehlender Beziehungen nicht zu fehlenden Attributwerten, sondern zum Fehlen von Paaren in einem Set. Die stärker prozedurale, satzorientierte Form der Anfrageformulierung verlagert Probleme mit der Interpretation fehlender Attributwerte in Sätzen zum großen Teil in die Anwendungsprogrammierung. Bei manchen CODASYL-Systemen wurde wegen der engen Verbindung zur Programmiersprache COBOL die spezielle figurative Konstante *low value* für die Darstellung

fehlender Werte in einer Datenbank gewählt. Rechenregeln für diesen Wert ergeben sich dann direkt aus den Regeln für die Sprache COBOL.

Schon bei Einführung des relationalen Datenmodells durch E.F. Codd ([Cod70], [Cod72a,b]) werden undefinierte Werte erwähnt und zwar im Zusammenhang mit dem Schlüsselkonzept ([Cod72a], S. 37):

... The usual operational distinction between the primary key and other candidate keys (if any) is that no tuple is allowed to have an undefined value for any of the primary key components, whereas any other components may have an undefined value.

Weitergehende Festlegungen bezüglich undefinierter Werte, insbesondere im Hinblick auf ihre Berücksichtigung bei der Auswertung von Anfragen, finden sich in diesen grundlegenden Arbeiten allerdings nicht. Erst in [Cod75] und dann ausführlicher in [Cod79] werden Vorschläge gemacht, wie die Operatoren der Relationenalgebra abgeändert werden können, um undefinierte Werte bei festgelegtem Grund für die Undefiniertheit geeignet zu berücksichtigen. Als Grund wird dabei ausschließlich "value at present unknown" angenommen, d.h. das Vorkommen eines undefinierten Attributwertes bedeutet, daß das Vorhandensein eines Wertes bekannt ist, der konkrete Wert aber nicht.

In zeitlicher Nähe und teilweise mit engem Bezug zu den Arbeiten von Codd erschienen eine ganze Reihe von Arbeiten zu "unvollständiger Information" ([Lip76], [Lip77], [Jae78], [Lip79], [Grt80], [IL81]), "Nullwerten" ([Grt77], [Bis81], [Vas79]) und "partiellen Werten" ([Grt79]), die sich mit undefinierten Attributwerten im relationalen Datenmodell oder damit eng verwandten Datenmodellen beschäftigen. Bis auf [Vas79] liegt allen Arbeiten die oben angegebene Bedeutung "Wert unbekannt" mit verschiedenen Varianten des Wissens über die jeweilige Menge möglicher Werte zugrunde. In [Vas79] wird zusätzlich die Bedeutung "Wert undefiniert" betrachtet, allerdings kann jedes Vorkommen eines fehlenden Wertes nur genau eine der beiden Bedeutungen tragen.

In den Arbeiten von W. Lipski wird als mathematisches Modell einer Datenbank ein sogenanntes Informationssystem mit Objekten betrachtet und untersucht, wie Anfragen interpretiert werden können, wenn statt gewöhnlicher Werte endliche Wertemengen in Attributen von Objekten erlaubt sind. Durch diese Teilmengen von Wertebereichen wird die Menge möglicher Werte an den betreffenden Stellen eingeschränkt. Es werden zwei Schranken für Antworten auf Anfragen betrachtet: Eine obere Schranke mit allen Objekten, die möglicherweise in der Antwort enthalten sind, und eine untere Schranke mit allen Objekten, die mit Sicherheit zur Antwort gehören. Ein Informationssystem entspricht einer relationalen Datenbank mit nur einer Relation. Die betrachteten Anfragen sind eingeschränkt auf Selektionsausdrücke, d.h. Boolesche Ausdrücke über einfachen Vergleichsausdrücken.

J. Grant ([Grt80]) greift einige Ideen von Lipski auf und untersucht Intervalle numerischer Werte bzw. unvollständige Zeichenreihen für allgemeine relationale Datenbanken. Er gibt gesicherte und mögliche Versionen für die Operatoren der Relationenalgebra an, untersucht allerdings keine allgemeinen Ausdrücke. Eine Umsetzung der Vorschläge von Lipski bei der Erweiterung einer konkreten Anfragesprache (QbE) ist in [WN88] beschrieben.

J. Biskup geht in [Bis81] und später auch in [Bis84] von endlichen Wertebereichen für alle Attribute aus und schlägt zwei Typen neuer Werte für alle Wertebereiche

vor: "∃" steht für genau einen unbekannten Wert aus dem zugehörigen Wertebereich; "∀" trägt die Information, daß an der betreffenden Stelle jeder Wert des zugehörigen Wertebereichs einzusetzen ist. Für derartig geänderte Relationen wird eine partielle Ordnung definiert und mit Bezug auf diese Ordnung festgelegt, wann Operationen korrekt und vollständig sind. Es wird nur ein Teil der Operatoren der Relationenalgebra betrachtet, insbesondere werden auch keine Ausdrücke mit mehreren Operatoren, d.h. komplexere Anfragen, untersucht. Die Definitionen für die Korrektheit und Vollständigkeit von Operationen finden sich in etwas anderer Form bei der Festlegung verschiedener Antwortformen im dritten Kapitel dieser Arbeit wieder.

Im Ansatz von Y. Vassiliou ([Vas79]) werden Anfragen an eine relationale Datenbank zunächst allgemein als Funktionen eingeführt. Um ihnen eine Bedeutung im Fall unbekannter oder definitiv fehlender Werte zuzuordnen, wird eine denotationelle Semantik angegeben. Dazu werden Wertebereiche um top- und bottom-Elemente erweitert, durch die die beiden Typen fehlender Attributwerte Berücksichtigung finden. Anschließend können mit den hinzugefügten Elementen auf Wertebereichen partiell definierte Funktionen zu totalen, stetigen Funktionen erweitert werden. Damit läßt sich die Semantik von Anfragen mit Hilfe bekannter Mittel angeben. Sind keine top-Elemente in einer Datenbank vorhanden, dann entspricht die "true version" einer Anfrage dem, was wir in Kapitel 3 eine gesicherte uni-Antwort im Fall von ω-Relationen nennen. Mit der Semantikdefinition hat man allerdings noch kein effizientes Verfahren zur Berechnung von Anfragen. In [Vas79] werden lediglich Selektionsausdrücke betrachtet. Für Anfragen mit solchen Ausdrücken wird eine Methode zur Auswertung von Anfragen vorgestellt, die über allgemeine disjunktive Normalformen führt und daher im allgemeinen nicht effizient ist. Auch die in [Sik81] vorgeschlagenen Verbesserungen können nicht zu einer effizienten und vollständigen Methode führen, da wir es im Kern mit dem Tautologieproblem bei Booleschen Ausdrücken zu tun haben.

In [Cod79] bilden die Darstellung und Behandlung unbekannter Attributwerte (Nullwerte) einen Schwerpunkt bei den Vorschlägen zur Erweiterung des relationalen Datenmodells. Es wird - wie oben schon erwähnt - angenommen, daß an einer Stelle mit einem undefinierten Wert genau ein Wert stehen soll, der Wert aber nicht bekannt ist. Für die Operatoren der Relationenalgebra werden jeweils zwei Definitionen angegeben: eine *sure*-Version und eine *maybe*-Version. Durch die sure-Versionen soll garantiert werden, daß in Antworten nur Tupel vorkommen, die sich unabhängig von den Werten, die unbekannte Attributwerte repräsentieren, qualifizieren. Bei Wahl der maybe-Versionen werden auch Tupel in Antworten aufgenommen, für die nicht ausgeschlossen werden kann, daß sie sich mit Sicherheit qualifizieren.

Grant kritisiert in [Grt77] die Vorschläge von Codd, indem er auf Probleme des verwendeten "Nullersetzungsprinzips" verweist (siehe hierzu Kapitel 4, Abschnitt 4.1). Im wesentlichen ist damit die Problematik der Tautologieerkennung gemeint, die mit unbekannten Werten verknüpft ist und durch das Nullersetzungsprinzip nicht voll erfaßt wird. Da das Tautologieproblem bekanntermaßen schon für Boolesche Ausdrücke ein coNP-vollständiges Problem ist ([GJ79]), können die in [Grt77] gemachten Vorschläge für gewöhnliche Anfragesprachen im allgemeinen nur auf Kosten der Effizienz zu vollständigen Antworten führen.

Biskup setzt sich in [Bis83] grundsätzlicher mit dem Vorgehen von Codd auseinander. Er schlägt eine Erweiterung der Relationen um ein Attribut STATUS vor,

in dem für jedes Tupel vermerkt wird, ob es *mit Sicherheit* oder nur *möglicherweise* zur Relation gehört. Auf der Basis dieser Darstellung und einer etwas abgeänderten Bedeutung der Vorkommen von Nullwerten werden in [Bis83] Definitionen für eine vollständige Menge von Algebraoperationen angegeben. Es wird für diese Operationen gezeigt, daß sie korrekt und vollständig sind gemäß einer geeigneten Anpassung dieser schon in [Bis81] eingeführten Eigenschaften an die erweiterten Relationen. Wir werden in Kapitel 4 und 5 auf diesen Ansatz zurückkommen. Es sei hier nur schon darauf hingewiesen, daß die Abänderung der Bedeutung aller Vorkommen von Nullwerten in Relationen zu einem nicht unwesentlichen Unterschied zum Vorschlag von Codd führt, wenn keine Beschränkungen für die Vorkommen von Nullwerten gelten sollen.

Wesentliche Aussagen im Hinblick auf prinzipielle Fragen im Zusammenhang mit der Darstellung und Behandlung unbekannter Attributwerte sind von T. Imielinski und W. Lipski in [IL81] und [IL84], der ausführlichen Fassung von [IL81], gemacht worden. In diesen Arbeiten wird gezeigt, daß die Verwendung eines einzigen Symbols für das Markieren von Vorkommen unbekannter Attributwerte nicht ausreicht, um alle Operatoren der Relationenalgebra im Hinblick auf eine bestimmte Antwortform geeignet definieren zu können. Als Antwortform wird dabei die totale gesicherte Antwort zugrunde gelegt. In Antworten dieser Form sind alle Tupel enthalten, die total definiert sind, also keine Nullwerte enthalten, und für jede Datenbank, die durch eine Datenbank mit Nullwerten beschrieben wird, in der Antwort auf die betrachtete Anfrage enthalten sind. Als Anfragen werden hierbei konkret nur Ausdrücke der Relationenalgebra in etwas verallgemeinerter Form untersucht. Operationsdefinitionen werden als geeignet betrachtet, wenn sie stets Ergebnisse liefern, in denen keine Information verlorengegangen ist, die zur Bestimmung gesicherter totaler Antworten für einen Ausdruck insgesamt notwendig ist. Ein wesentliches Ergebnis von [IL84] ist, daß mit Verwendung eines einzigen Symbols für Nullwerte die gewünschten Ergebnisse nur für Ausdrücke mit Projektionen und/oder Selektionen erhalten werden können. Werden dagegen für alle Vorkommen unbekannter Attributwerte in den Relationen einer Datenbank voneinander verschiedene Symbole benutzt, dann können Definitionen für die Operatoren Projektion, positive Selektion, Vereinigung und Join angegeben werden, so daß Ausdrücke mit diesen Operatoren in der gewünschten Weise ausgewertet werden können. Wir werden hierauf in Kapitel 6 genauer eingehen.

Um gesicherte totale Antworten vollständig bestimmen zu können, schlagen Imielinski und Lipski die Erweiterung von Relationen um ein Attribut COND vor, in dem Bedingungen zu den verschiedenen markierten Vorkommen von Nullwerten abgelegt werden können. Diese Bedingungen geben an, unter welchen Voraussetzungen das entsprechende Tupel in der Relation enthalten ist. Sie stellen daher eine Art Verfeinerung der Angaben in den oben erwähnten COND-Attributen dar. Jede Bedingung ist ein Boolescher Ausdruck über Vergleichsausdrücken, in denen Nullwertsymbole als Variable auftreten können, und gibt somit an, für welche Werte der zugehörigen Wertebereiche ein Tupel in einer Relation enthalten ist. Tupel sind mit Sicherheit in einer Relation enthalten, wenn ihr Bedingungsausdruck eine Tautologie darstellt. Auch hier ist somit das angesprochene Tautologieproblem zu beachten.

G. Grahne verallgemeinert in [Gra91] diese sogenannten C-Tabellen, indem er tupelunabhängige Bedingungen erlaubt. Daneben führt er auch einen Fixpunktopera-

tor auf unvollständigen Datenbanken ein, der in Anfragen verwendet werden darf, und untersucht die Komplexität sowohl von Anfragen als auch von Änderungsoperationen. Als Antwortformen verwendet er die gesicherte totale Antwort und die mögliche Antwort aus [IL84].

In den erwähnten Arbeiten werden zwar bei den Definitionen der Operationen meist Korrektheit und Vollständigkeit diskutiert (allerdings auf unterschiedlichem formalen Niveau), eine genauere Untersuchung der Resultate komplexer Ausdrücke findet sich aber bis auf [IL84] und [Gra91] nicht. Auch werden keine vollständigen kalkülorientierten Anfragesprachen betrachtet.

Ein Ansatz, das Tautologieproblem zu vermeiden und damit zu effizienten Verfahren für die Bestimmung korrekter und vollständiger Antworten auf Anfragen für eine vollständige Anfragesprache zu kommen, wird von C. Zaniolo in [Zan82] und ausführlicher in [Zan84] vorgestellt. Die Idee besteht darin, die Bedeutung der Vorkommen fehlender Attributwerte abzuändern in "keine Information vorhanden (no information)". Dies ist genau die Bedeutung, die den "Nullwerten" in der de facto Standardsprache SQL ([ISO89], [ISO92]) zugrunde liegt, und einige der Vorschläge von Zaniolo passen auch zu den Definitionen von SQL-Prädikaten und -Klauseln. Auf Probleme, die mit diesen Vorschlägen verbunden sind, wurde von A.M. Keller hingewiesen ([Kel86]). Wir diskutieren sie genauer in den Kapiteln 7 und 8.

Die schon angesprochenen Komplexitätsprobleme im Zusammenhang mit unbekannten Attributwerten werden von M.Y. Vardi in [Var85] untersucht. Gewöhnliche Datenbanken ohne fehlende Werte werden physische Datenbanken genannt und als Modelle logischer Datenbanken betrachtet. Auf die logikorientierte Sicht von Datenbanken gehen wir unten und in Kapitel 6 näher ein. Hier wollen wir nur vermerken, daß für eine logische Anfragesprache mit der Ausdruckskraft der Relationenalgebra gezeigt wird, daß die Berechnung einer bestimmten Form gesicherter Antworten ein co-NP-vollständiges Problem ist und damit zur gleichen Klasse wie das Tautologieproblem gehört.

S. Abiteboul, P. Kanellakis und G. Grahne führen in [AKG87] sowie ausführlicher in [AKG91] Komplexitätsuntersuchungen für verschiedene Formen der Darstellung unbekannter Werte durch. Unter den betrachten Varianten sind auch die von Imielinski und Lipski eingeführten Relationen mit COND-Attribut sowie Erweiterungen davon. Es werden verschiedene Anfrage/Antwort-Kombinationen untersucht, darunter auch gesicherte und mögliche Antworten. Es zeigt sich, daß die meisten Fragestellungen in einer hohen Komplexitätsklasse bleiben, auch wenn Beschränkungen bezüglich der in Datenbanken repräsentierbaren Information vorgenommen werden.

Grundlegend für die schon erwähnte logikorientierte Sichtweise relationaler Datenbanken waren die Arbeiten von J.M. Nicolas und H. Gallaire ([NG78]) und R. Reiter ([Rei78a,b]). Diese Sichtweise ermöglicht es auf einfache Weise, Unvollständigkeit von Wissen über Werte in einer Datenbank zu modellieren, indem etwa Fakten durch Disjunktionen festgelegt werden. In [Rei84] wird gezeigt, wie relationale Datenbanken mit unbekannten Werten, deren Vorkommen durch Variable markiert sind, als spezielle Theorien der Logik erster Stufe gesehen werden können. Gegenüber Theorien für Datenbanken ohne fehlende Werte sind lediglich einige der Axiome wegzulassen, durch die der Inhalt einer Datenbank festgelegt wird. Bei der Ableitung von Fakten für die Ermittlung von Antworten zeigt sich aber, daß im

Prinzip die gleichen Schwierigkeiten auftreten wie bei der üblichen modelltheoretischen Sichtweise. Dies ist wegen der in [Rei84] gezeigten Äquivalenz beider Sichtweisen auch nicht weiter verwunderlich. Wir werden darauf in Kapitel 6 näher eingehen.

Unvollständige Information kann für Datenbanken auch im Zusammenhang mit der Verteilung von Daten auftreten, ohne daß Werte in Relationen fehlen. Richtet sich eine Anfrage an mehrere Datenbanken, in denen Information zum gleichen Sachverhalt gespeichert ist, dann kann aufgrund unterschiedlicher Information in den einzelnen Datenbanken eine Antwort unvollständig bleiben. Auch in solchen Fällen kann von gesicherter und möglicher Information wie bei den von Lipski betrachteten oberen und unteren Schranken gesprochen werden. In [Lib95a] wird eine Systematik entwickelt für unterschiedliche Annahmen bezüglich der Struktur und der Zusammenhänge zwischen unterer und oberer Schranke. Die vorgestellte Theorie zielt in eine andere Richtung als die bisher erwähnten Arbeiten, da sie andere Formen fehlender Information betrachtet. Es gibt aber Gemeinsamkeiten bei der Behandlung der Ordnung von partiellen Relation gemäß ihrem Informationsgehalt, wo den Überdeckungen und engen Ausdehnungen (siehe Kapitel 2) entsprechende Ordnungen verwendet werden.

Auf der Basis dieser partiellen Ordnungen beschäftigt sich Libkin in [Lib95b] auch mit Datenbanken, die unvollständige Information repräsentieren. Vervollständigungen, wie wir sie in Kapitel 4 betrachten, werden dabei nicht diskutiert. Es wird ein allgemeines Sprachkonzept vorgestellt, das komplexe Typen berücksichtigt und mit dessen Hilfe sich die Semantik anderer Sprachen, wie zum Beispiel die spezielle Algebra von Zaniolo, definieren läßt. Ziel ist weniger die Entwickling konkreter Algorithmen für die Anfragebearbeitung als ein allgemeines Werkzeug für die Definition der Semantik von Sprachen für Datenbanken mit unvollständiger Information.

Es gibt eine große Zahl weiterer Arbeiten, die sich mit partiell definierten Relationen in Datenbanken unter den unterschiedlichsten Annahmen bezüglich der Anzahl verschiedener Formen von "Nullwerten" und ihrer Bedeutung befassen (z.B. [Gol85], [Cod86], [Cod87], [GZ88], [Mor90]). Dabei wird Unvollständigkeit des Wissens nicht nur auf Werteniveau, sondern auch auf der Ebene von Tupeln und Relationen betrachtet ([LS90], [LS91], [LZ91],[OO93]). Unterschiedliche Bedeutungen von fehlenden Werten in einer Datenbank werden häufig durch Einführung neuer mehrwertiger Logiken berücksichtigt ([GZ88], [Ges90], [Ges91], [GaMo90], [Yue91]), was zu Problemen führen kann, wenn Anwender die Auswirkungen vollständig verstehen müssen, um Antworten auf Anfragen richtig interpretieren zu können. Betrachtungen fehlender Attributwerte in Erweiterungen des relationalen Datenmodells, in denen Relationen mit komplexen Objekten, wie z.B. geschachtelte Relationen, erlaubt werden, haben sich als schwierig erwiesen ([RKS89], [RKS91], [TG92], [LeLo93]) und führen zu unhandlichen Definitionen für die Operatoren.

Wir wollen uns in dieser Arbeit mit einigen grundsätzlichen Problemen beschäftigen, die in den oben vorgestellten Arbeiten nicht oder nicht erschöpfend behandelt werden und mit einer systematischen Ausdehnung der Konstrukte und Sprachen des relationalen Datenmodells von total definierten auf partiell definierte Relationen unmittelbar verbunden sind. Dabei beschränken wir uns auf das klassische Datenmodell mit flachen Relationen, d.h. ohne komplexe Objekte als Attributwerte. Wir

betrachten im wesentlichen zwei mögliche Bedeutungen fehlender Werte in Datenbanken: "Wert existiert, ist aber nicht bekannt" und "es ist keine Information zum Wert vorhanden". Beide Bedeutungen treten häufig in Anwendungen auf. Die erste Bedeutung ist die am meisten untersuchte; die zweite Bedeutung liegt fehlenden Werten in der Sprache SQL zugrunde und ist daher von besonderem praktischen Interesse. Wir wollen bei unseren Untersuchungen keine Vermischung dieser Bedeutungen in Datenbanken annehmen, sondern sie getrennt betrachten. Außerdem schränken wir in einem Kapitel die zweite Bedeutung weiter ein auf "Wert existiert, ist aber nicht bekannt" oder "Wert ist definitiv nicht vorhanden", d.h. eine Alternativkombination der ersten Bedeutung und einer Bedeutung, die vollständiges Wissen darstellt.

Die erste Frage, die sich beim Übergang von Datenbanken mit totalen zu Datenbanken mit partiellen Relationen stellt, ist die Frage nach der Semantik solcher partiell definierten Relationen. Für Relationen, in denen undefinierte Werte als unbekannte Werte interpretiert werden, ist folgende Festlegung naheliegend: Die Semantik einer Relation mit unbekannten Attributwerten ist gegeben durch die Menge aller totalen Relationen, die bei Ersetzung der fehlenden Werte durch definierte gewöhnliche Werte erhalten werden. Dabei ist die Ersetzungsvorschrift in Abhängigkeit von der Realisierungsform partieller Relationen bestimmt. Für die zweite Bedeutung undefinierter Attributwerte kann die Semantik partieller Relationen nicht in gleicher Weise auf die Semantik totaler Relationen zurückgeführt werden. Mit der vorgenommenen Differenzierung bleibt das Problem, die Semantik von Relationen festzulegen, in denen Werte fehlen. Diese Relationen repäsentieren zwar vollständige Information, es treten aber ähnliche Schwierigkeiten auf, wie man sie von Universalrelationen her kennt ([MUV84]).

Mit einer über - eventuell unendliche - Mengen festgelegten Semantik von partiellen Relationen und damit von Datenbanken mit solchen Relationen (kurz partiellen Datenbanken) ergibt sich unmittelbar eine Semantik für Anfragen: Eine Anfrage ist in in allen möglichen Datenbanken zu einer partiellen Datenbank, die durch die Semantik der partiellen Relationen bestimmt sind, auszuwerten. Die Menge aller (möglichen) Antworten, die auf diese Weise erhalten wird, ist die Semantik der Anfrage an die partielle Datenbank. Mit dieser Definition ist natürlich im allgemeinen ein Darstellungs- und Berechnungsproblem verbunden, da die Anzahl der möglichen Antworten unendlich oder doch zumindest sehr groß sein kann.

Bei der Suche nach der Lösung dieses Darstellungs- und Berechnungsproblems scheint es ratsam zu beachten, was in folgendem Zitat aus dem Buch *The Logic of Questions and Answers* von N.D. Belnap und T.B. Steel angesprochen wird ([BS76], S. 2):

The meaning of a question addressed to a query system is not to be identified with how the system processes the query (and is not to be identified with a program at any level), but rather is to be identified with the range of answers that the question permits. That is, for a query system and a user to agree on the meaning of a question is for there to be agreement as to what counts as an answer to the question, regardless of how, or if, any answer is produced.

Gemäß dieser Forderung sollten sich Lösungen des Darstellungsproblems nicht allein auf die Definition spezieller Operatoren oder Logiken abstützen, sondern davon

unabhängig interpretierbare und den Anwendungsbedürfnissen angepaßte Antwortformen anbieten.

Eine in dieser Hinsicht geeignete Antwortform wurde oben schon erwähnt: die gesicherte totale Antwort. Wir fügen dieser Antwortform weitere Varianten gesicherter Antwortformen hinzu. Dabei verstehen wir unter gesicherten Antworten ganz allgemein Antworten, die nur Information enthalten, die in allen zugehörigen möglichen Antworten gültig ist. Mit solchen Formen läßt sich die im allgemeinen unendliche Menge möglicher Antworten approximieren.

Die Ermittlung vollständiger Antworten ist für jede der betrachteten Antwortformen mit dem bekannten Tautologieproblem für Boolesche Ausdrücke verknüpft. Vollständige Antworten sind daher bei Datenbanken von realistischer Größe im allgemeinen nicht ermittelbar. Bei der Suche nach Verfahren sollte man sich vielleicht (unter Beachtung des Zitats von oben) von folgender Überzeugung leiten lassen, die A.M. Keller und M. Winslett in [KW85] geäußert haben:

We believe that the best answer to a query is a function of the computational power that the user is willing to expend for that purpose.

In [Cod79] verwendet Codd eine dreiwertige Logik, um effizient Antworten berechnen zu können. Er gibt allerdings nur die Funktionstabellen für die drei Junktoren $\wedge$, $\vee$ und $\neg$ an und sagt nichts Näheres zu dieser Logik; insbesondere macht er keine Angaben zu eventuell gültigen Äquivalenzen für andere Junktoren. Es gibt eine ganze Reihe von Vorschlägen für dreiwertige Logiken; die von Codd angegebenen Funktionstabellen passen zum Beispiel sowohl zu der 1920 von Lukasiewicz als auch zu der 1938 von Kleene angegebenen dreiwertigen Logik. Aus der folgenden Bemerkung von A.A. Sinowjew zu mehrwertigen Logiken allgemein kann sowohl ein eventuelles Anwendungsproblem für dreiwertige Logiken gefolgert als auch eine Mahnung zum durchdachten Umgang mit ihr abgelesen werden.

A.A. Sinowjew in *Über mehrwertige Logik* ([Sin68], S. 62):

Wenn man davon spricht, daß in der mehrwertigen Logik nicht alle Gesetze der zweiwertigen Logik erhalten bleiben, so führt dies manchmal bei Menschen, denen die mehrwertige Logik nicht genügend vertraut ist, zu einem mystischen Anstaunen der Geheimnisse "der Natur" oder aber umgekehrt zu dem Bewußtsein, hier liege eine offensichtliche Unsinnigkeit vor. In Wirklichkeit handelt es sich jedoch um einen ziemlich trivialen Fakt, wenn genau feststeht, worüber man spricht (über welche "Gesetze" und unter welchen Bedingungen).

Ein einfaches Beispiel macht die Probleme deutlich, mit denen man es bei Anwendung der dreiwertigen Logik für die Auswertung von Anfragen zu tun bekommt. Sei in einer Anfrage unter Verwendung der Attribute A,B und C die Bedingung A = B AND B = C angegeben. Ist der Wert für B unbekannt und wird aus diesem Grund den beiden Vergleichsausdrücken der (dritte) Wert *unbekannt* (oder *undefiniert*) zugeordnet, dann ist das Ergebnis für die Logiken von Lukasiewicz und Kleene der Wert *unbekannt* (*undefiniert*) unabhängig von den Werten in A und C. Ersetzen wir dagegen die Bedingung durch die in der zweiwertigen Booleschen Logik äquivalente Form A = B AND A = C, dann erhalten wir *falsch* als Ergebnis, wenn die Werte in A und C verschieden voneinander sind. Für die Umkehrung gilt entsprechendes. Dieses Problem ist insbesondere bei der Optimierung von Ausdrücken zu beachten.

Aus dem Beispiel folgt direkt (verwende die negierte Form der Bedingung in einer einfachen Anfrage), daß die Anwendung einer dreiwertigen Logik zur Unvollständigkeit gesicherter Antworten führen kann. Trotz dieses Problems und der Vorsicht, die bei Äquivalenzen geboten ist, stützen sich die von uns vorgeschlagenen Verfahren auf diese Logik. Die Gründe hierfür sind, daß wir mit einer einheitlich zugrundeliegenden Auswertungsmethode die Vergleichbarkeit aller Verfahren gewährleisten und daß Effizienz wie bei der gewöhnlichen Auswertung garantiert ist. Eventuelle Verbesserungen im Hinblick auf die Vollständigkeit von Antworten können an den entsprechenden Stellen nachträglich vorgenommen werden. Wir gehen darauf aber nur am Rande ein.

Als Anfragesprachen betrachten wir die Relationenalgebra und den tupelorientierten Relationenkalkül sowie im letzten Kapitel auch SQL. Wir stellen die Auswertungsverfahren für die Relationenalgebra und den Kalkül einander gegenüber, um dadurch Gemeinsamkeiten und Unterschiede bei prozeduralen und deskriptiven Sprachen zeigen zu können. Außerdem spielen Konzepte beider Sprachen bei der Weiterentwicklung von Sprachvorschlägen eine wichtige Rolle ([ISO92]).

In den nachfolgenden fünf Kapiteln beschäftigen wir uns ausschließlich mit der Bedeutung "Attributwert existiert, ist aber nicht bekannt" oder kurz "Wert unbekannt" für fehlende Attributwerte. Dabei betrachten wir zwei unterschiedliche Erweiterungen gewöhnlicher Relationen für die Darstellung unbekannter Werte: ω-Relationen und V-Relationen. In ω-Relationen dient ein einziges ausgezeichnetes Symbol dazu, Vorkommen unbekannter Attributwerte anzuzeigen; in V-Relationen stehen Variable zur Verfügung, um jedes Vorkommen eines unbekannten Wertes in der Datenbank eindeutig zu kennzeichnen. Die Beschränkung auf ein Symbol bei ω-Relationen legt es nahe, zwei verschiedene Wege zuzulassen, um durch Ersetzungen unbekannter Werte zu totalen Relationen im Sinne der oben angesprochenen Definition der Semantik partieller Relationen zu gelangen. Wir untersuchen diese beiden Möglichkeiten getrennt in den Kapiteln 4 und 5.

In Kapitel 2 werden zunächst die grundlegenden Begriffe im Zusammenhang mit ω- und V-Relationen eingeführt. Dabei können wir uns zum großen Teil an der Literatur orientieren. Zusätzlich diskutieren wir einige Eigenschaften der erweiterten Relationsformen insbesondere im Hinblick auf Operationen zum Einfügen, Ändern und Löschen von Tupeln.

Im dritten Kapitel stehen verschiedene Formen von Antworten im Mittelpunkt. Wir diskutieren Antworten, die gesicherte Information repräsentieren, und Antworten, die möglicherweise gültige Information beinhalten. Dabei zeigt sich, daß sowohl für Datenbanken mit ω-Relationen als auch für Datenbanken mit V-Relationen mehrere Möglichkeiten zur Approximation möglicher Antworten durch gesicherte Antworten in Betracht gezogen werden sollten. Die Antwortformen sind unabhängig von konkreten Anfragesprachen definiert; es wird lediglich angenommen, daß Anfragen als Funktionen gesehen werden können, die Datenbanken in Relationen eines festen Typs abbilden. Als Antwortrelationen sind dabei insbesondere auch ω-Relationen und V-Relationen mit erweiterten Variablenmengen zugelassen. Für jede Antwortform wird angegeben, welche Information durch Antworten repräsentiert wird.

Die Berechnung gesicherter Antworten für den Fall, daß die Semantik von ω-Relationen über Vervollständigungen definiert ist, steht im Mittelpunkt des vierten

Kapitels. Vervollständigungen sind totale Relationen, die durch das Ersetzen jedes Vorkommens von ω in einer ω-Relation durch genau einen Wert erhalten werden. Wir diskutieren zunächst zwei Vorschläge aus der Literatur und geben dann sowohl für den tupelorientierten Relationenkalkül als auch für die Relationenalgebra mehrere Verfahren an, die gesicherte Antworten garantieren. Dabei zeigen wir die Korrektheit der Verfahren und betrachten für zwei Verfahren auch die klassische Äquivalenz der beiden Sprachen, die sich unter ihrer Verwendung ergibt.

Kann jedes Vorkommen von ω in einer ω-Relation durch beliebig viele gewöhnliche Werte ersetzt werden, dann erhält man sogenannte enge Ausdehnungen von ω-Relationen. Nach einer Diskussion des Vorschlags von Biskup in [Bis83], der sich unter Annahme dieser Semantikdefinition mit Operationen für partielle Relationen befaßt, zeigen wir zunächst, daß gesicherte Antworten im Fall unendlicher Wertebereiche nicht berechenbar sind. Anschließend diskutieren wir auch hier verschiedene Auswertungsverfahren für den Relationenkalkül und die Relationenalgebra.

V-Relationen sind Gegenstand des sechsten Kapitels. Wir übertragen Ideen aus den beiden vorangehenden Kapiteln auf Datenbanken mit V-Relationen und kommen so zu effizienten Verfahren für die Bestimmung gesicherter (unvollständiger) Antworten auf beliebige Algebraanfragen. Dabei ergibt sich eine wesentliche Aussage von [IL84] zu monotonen Anfragen als Korollar. Die in [IL84] eingeführten C-Relationen verwenden wir anschließend, um für Algebraanfragen Algorithmen zur Berechnung gesicherter Antworten anzugeben. Mit Hilfe einer einfachen Änderung der Interpretationsvorschrift kann die Vorgehensweise auf Kalkülanfragen übertragen werden. Da bei beweistheoretischen Betrachtungen von Datenbanken meist Annahmen zugrunde liegen, die zur modelltheoretischen Sichtweise mit V-Relationen passen, vergleichen wir im sechsten Kapitel unsere Methoden mit beweistheoretischen Auswertungsverfahren. Es zeigt sich, daß eine der Methoden mit den gleichen Kosten dasselbe Ergebnis liefert wie das in [Rei86] vorgeschlagene Verfahren. Zum Schluß diskutieren wir noch Probleme mit einem Ansatz, der die intuitionistische Logik verwendet, um zu korrekten und vollständigen Ergebnissen zu gelangen.

Im siebten Kapitel wenden wir uns der zweiten von uns betrachteten Bedeutung fehlender Attributwerte zu. Wir demonstrieren an Hand von Beispielen einige Schwierigkeiten, die mit der Definition der Semantik von partiellen Datenbanken und von Anfragesprachen verbunden sind, wenn fehlende Werte alternativ zu "Wert existiert, ist aber nicht bekannt" die Bedeutung haben können, daß an der betreffenden Stelle auch für den modellierten Sachverhalt kein Wert vorhanden ist. Anschließend schlagen wir eine Semantikdefinition für Datenbanken vor und diskutieren Effizienzprobleme bei der Bestimmung gesicherter Antworten, wobei wir ein Verfahren aus der Literatur einbeziehen. Einige Bemerkungen zu dem oben schon erwähnten Vorschlag von Zaniolo bilden den Abschluß.

Im letzten Kapitel setzen wir uns mit der Nullwertproblematik der Sprache SQL auseinander, in der fehlende Werte das Fehlen jeder Information zu einem Wert bedeuten. Nach einer Diskussion der Ursachen dieser Problematik definieren wir eine Form gesicherter Antworten für SQL-Anfragen, die ohne Änderung der SQL-Semantik gesicherte Information in Antworten garantiert. Für diese Antwortform geben wir ein Verfahren zum Ergänzen und Modifizieren von SQL-Anfragen an, durch das abgesichert werden kann, daß Antworten stets gesichert im zuvor definierten Sinn sind. Es kann auch zur Bestimmung sogenannter potentieller Antworten verwendet werden.

2 Modellierung unvollständiger Information

Bei der Interpretation von Daten, die in einer Datenbank gespeichert sind, ist zwischen der sogenannten *äußeren Semantik* und der *Modellsemantik* zu unterscheiden. Die äußere Semantik ergibt sich durch die Verknüpfung von gespeicherten Daten mit dem Kontext des Betrachters der Daten. Daten werden vom Betrachter in Abhängigkeit vom Datenbankschema bzw. den Kenntnissen, die der Betrachter davon hat, mit (materiellen oder ideellen) *Gegenständen* assoziiert, deren interessierende und beobachtbare oder angenommene Eigenschaften durch die Daten repräsentiert werden. Enthält z.B. ein relationales DB-Schema einen Relationstyp **PERSON** mit den Attributen NACHNAME, VORNAME und GEBURTSTAG, dann wird ein Benutzer mit dem Wert "28.2." im Attribut GEBURTSTAG einen ganz bestimmten Tag im Jahr assoziieren. Dies wird allein durch die gewählten Bezeichner und den Wert veranlasst.

Bemerkung: Die Verwendung des Begriffs "Gegenstand" erfolgt hier im weitesten Sinn so, wie von A.A. Sinowjew in [Sin70], S. 30, angegeben:

Ein Gegenstand ist alles, was wahrgenommen, vorgestellt, benannt usw. werden kann, kurz gesagt, alles Beliebige.

Im Unterschied zur äußeren Semantik bestimmt sich die Modellsemantik der in einer Datenbank gespeicherten Daten allein aus der mit Konstrukten des verwendeten Datenmodells gebildeten Struktur der Daten sowie den Mitteln, die das Datenmodell für die Handhabung von Daten zur Verfügung stellt. Betrachten wir dazu das relationale Datenmodell in seiner ursprünglichen Form ([Cod70], [Cod72a,b]): Werte von Tupeln einer Relation sind hier uninterpretierte Elemente der den Attributen der Relation eindeutig zugeordneten Wertebereiche. Dies bedeutet insbesondere, daß die durch einen Zustand zu einem Datenbankschema gegebene Modellsemantik unter Abbildungen erhalten bleibt, die Bijektionen auf Wertebereichen darstellen. Eventuell auf einem Wertebereich vorhandene Ordnungsrelationen sind dabei anzupassen. Abgesehen von der sich durch Ordnungsrelationen ergebenden Information trägt daher jeder Wert nur die dem zugehörigen Wertebereich als Menge anhaftende Information der Unterscheidbarkeit von anderen Werten. Diese Eigenschaft wird auch "genericity property" genannt ([AKG91]).

Wird der Wert "28.2." von oben systematisch, d.h. an allen Stellen, an denen er vorkommt, durch einen neuen Wert, z.B. "30.2.", ersetzt und kommt dieser neue Wert an keiner anderen Stelle in dem betrachteten Datenbankzustand vor, dann ändert sich die modellinhärente Semantik des Zustands nicht.

Anfragesprachen, die zur modellinhärenten Semantik passen, sind deshalb so definiert, daß sich Antworten zu Anfragen unter den angesprochenen Bijektionen auf Wertebereichen nicht ändern. Dies bedeutet auch, daß Darstellungsformen von Werten bei der Auswertung von Anfragen nicht angesprochen und ausgenutzt werden können. Ein Beispiel für eine solche Ausnutzung ist in der folgenden Anfrage gegeben: "Gib die Daten aller Personen, in deren Nachnamen mehr als einmal der Buchstabe e vorkommt".

Bemerkung: Konkrete Anfragesprachen verfügen durchaus über Sprachkonstrukte, die das Abfragen von Eigenschaften auf dem Niveau der Zeichenreihen, durch die Werte repräsentiert werden, erlauben. Wir wollen uns hier aber auf das ursprüngliche

Modell beschränken, da durch derartige Konstrukte keine wesentlichen Problemstellungen im Zusammenhang mit dem in dieser Arbeit behandelten Thema hinzukommen.

Die Modellierung von Gegenständen durch Kombinationen von Mengenelementen setzt die Homogenität des abzubildenden Wissens über die Gegenstände voraus, d.h. die Art des Wissens bezüglich einzelner Werte muß gleich sein. Nicht immer ist diese Voraussetzung in der Praxis gegeben. Ein Attributwert kann aus irgendeinem Grund nicht bekannt sein, oder er existiert nicht, weil für den modellierten Gegenstand ein dem Attribut entsprechendes Merkmal nicht vorhanden ist. In solchen Fällen ist das Wissen über die Werte mit der einfachen mengenorientierten Auffassung von Wertebereichen nicht modellierbar. Sollen die wesentlichen Charakteristika des relationalen Datenmodells erhalten bleiben, kann zur Darstellung derartigen Wissens in Datenbankzuständen und zu seiner Nutzung bei der Auswertung eine Erweiterung von Wertebereichen um spezielle Werte vorgenommen werden. Dabei ist festzulegen, wie detailliert dieses Wissen in neuen Werten repräsentiert werden kann. Es ergeben sich hieraus insbesondere Bedingungen für update- (Einfüge-, Lösch- und Änderungs-) Operationen. Für die neu hinzukommenden Werte ist anzugeben, welche Information sie darstellen und welche Konsequenzen sich aus dieser Festlegung für die Interpretation von Zuständen ergeben. Die Definitionen von Operationen und die Angabe von Vorschriften zur Auswertung von Anfragen müssen so erfolgen, daß sie zu der angenommenen Information passen.

Zu einem relationalen Datenbankschema gehörende Integritätsbedingungen schränken im allgemeinen die Menge der Zustände, die aufgrund der vorhandenen Relationstypen und Wertebereiche für das Schema möglich sind, ein. Es werden nur solche Zustände als *erlaubt* angesehen, in denen alle angegebenen Integritätsbedingungen erfüllt sind. Bei Hinzunahme neuer Werte, deren Semantik sich von den übrigen Werten unterscheidet, muß die Definition erlaubter Datenbankzustände geeignet angepaßt werden. Für alle Integritätsbedingungen eines Datenbankschemas ist festzulegen, wann sie in Zuständen mit neuen Werten erfüllt sind. Außerdem ist zu beachten, daß durch das Ausnutzen von Integritätsbedingungen in bestimmten Fällen aus weniger präzisem Wissen ("es gibt einen Wert") auf detailliertes Wissen, d.h. einen gewöhnlichen Wert geschlossen werden kann. Gilt die funktionale Abhängigkeit $\beta \rightarrow A$, dann kann zum Beispiel bei Wertegleichheit zweier Tupel einer Datenbankrelation in den Attributen von β auf die Wertegleichheit (gewöhnliche und/oder unbekannte Werte) der beiden Tupel im Attribut A geschlossen werden, wenn die Werte in den Attributen von β gewöhnliche Werte sind. Auch andere Informationen, wie Aussagen über die Ungleichheit von fehlenden Attributwerten mit anderen (gewöhnlichen oder unbekannten) Attributwerten, können sich aus Integritätsbedingungen ergeben. Vorschläge hierzu werden zum Beispiel in [Lie79], [Vas80], [Gld81], [Gra84] und [Gra91] gemacht. Im folgenden wollen wir derartige ableitbare Informationen zu fehlenden Attributwerten nicht berücksichtigen. Ebenso wird partielles Wissen über Werte, etwa die Zugehörigkeit zu einem bestimmten Intervall oder einer anderen festen Teilmenge eines Wertebereichs, nicht betrachtet. Überlegungen hierzu finden sich zum Beispiel in [Grt79], [Lie79], [Lie81] und [Mor90]. Die einzige Form von Integritätsbedingung, die im folgenden eine Rolle spielt, ist die Schlüsseleigenschaft von Attributmengen.

In diesem und den vier folgenden Kapiteln wollen wir uns auf die am häufigsten in der Literatur betrachtete Bedeutung eines fehlenden Attributwertes konzentrieren, die meist wie folgt angegeben wird: Wert unbekannt, aber in der Realität

vorhanden. Versuchen wir, diese Charakterisierung etwas genauer zu fassen; dabei sollen die Begriffe *Merkmal* und *Eigenschaft* in der "Realität" den Konzepten *Attribut* und *Attributwert* oder kurz *Wert* im Datenmodell entsprechen:

Ein fehlender Attributwert in einem Tupel soll bedeuten, daß dem Gegenstand, der durch das Tupel modelliert wird, eine Eigenschaft in dem Merkmal, das dem Attribut entspricht, anhaftet, im DB-Zustand aber aus nicht weiter interessierenden Gründen nur die Existenz des Merkmals, nicht jedoch ein die Eigenschaft repräsentierender Wert bekannt ist.

Die Darstellung dieses Wissens mittels Erweiterung der Wertebereiche kann auf verschiedene Art und Weise erfolgen. Dabei können sich Unterschiede im Informationsgehalt von Zuständen ergeben, wenn Werte auch in Schlüsselattributen fehlen dürfen. Die beiden folgenden Formen der Erweiterung von Wertebereichen bieten sich an:

- Hinzunahme eines einzigen speziellen Symbols, das keinem Wertebereich zugeordnet ist und in keiner Vergleichsrelation (einschließlich "=") enthalten ist;
- Hinzunahme einer abzählbar unendlichen Menge neuer Symbole, die verschieden sind von allen Symbolen der gegebenen Wertebereiche und die in keiner Vergleichsrelation mit Ausnahme der Gleichheitsrelation enthalten sind (insbesondere darf dabei aus dem Nicht-Vorhandensein eines Wertepaares in der Gleichheitsrelation nicht auf die Ungleichheit der Werte geschlossen werden).

Betrachten wir zunächst die erste Alternative, die in der Literatur am häufigstens untersucht wurde und die für die Praxis aufgrund der einfachen Realisierbarkeit von besonderem Interesse ist. Anstatt der Menge aller Konstantensymbole eines Datenbankschemas ein einziges Symbol hinzuzufügen, kann auch jeder Wertebereich um ein spezielles Symbol erweitert werden. Diese Variante führt zu Zuständen mit dem gleichen Informationsgehalt wie bei Verwendung eines einzigen neuen Symbols, was sich unmittelbar aus der eindeutigen Zuordnung von Attributwerten zu Wertebereichen ergibt.

Sprechweise: Statt Datenbankschema, Datenbankzustand, usw. sagen wir im folgenden auch kurz DB-Schema, DB-Zustand, usw. oder nur Schema, Zustand,....

2.1 ω-Relationen

Sei ω als das erwähnte spezielle Symbol gewählt. Für ω soll weder die Gleichheit ω = ω noch irgendein Vergleich mit einem anderen Element eines Wertebereichs definiert sein. Dies entspricht der Annahme, daß zu einem unbekannten Attributwert außer der Information, daß er existiert, keine weitere Information vorliegt. Dem Vergleich ω = ω kann mit dieser Annahme nicht allgemein *wahr* als Wert zugeordnet werden, da verschiedene Vorkommen von ω in einem DB-Zustand unterschiedliche Werte repräsentieren können (von Ausnahmen wie einelementigen Wertebereichen einmal abgesehen). Datenbankrelationen, in denen das Vorkommen von ω erlaubt ist, wollen wir auch *ω-Relationen* nennen.

2.1.1 Update-Probleme

DB-Relationen sind im relationalen Datenmodell Mengen. Mit dem Zulassen unbekannter Werte in Attributen tritt folgendes Problem mit dem Mengenkonzept auf: Sei angenommen, daß unbekannte Werte auch in Primärschlüsselattributen

vorkommen dürfen, d.h. in Attributen, die zu einem gewählten Schlüssel gehören. Da dem Vergleich $\omega = \omega$ kein Wert zugeordnet werden darf, kann ein Tupel mit unbekannten Attributwerten, das in allen definierten Werten mit einem anderen Tupel übereinstimmt, nur syntaktisch als Duplikat dieses Tupels betrachtet werden, nicht aber semantisch, d.h. es kann von beiden Tupeln im allgemeinen nicht angenommen werden, daß sie den gleichen Gegenstand repräsentieren. Soll einer ω-Relation ein Tupel mit unbekannten Attributwerten hinzugefügt werden und ist in der ω-Relation schon ein syntaktisch identisches Tupel vorhanden, sind daher mehrere Vorgehensweisen denkbar:

- Es wird vom Mengenkonzept abgewichen und zu Tabellen übergegangen, in denen (syntaktische und semantische) Duplikate erlaubt sind.
- Die beiden Tupel werden als (syntaktische) Duplikate betrachtet und die ω-Relation ändert sich durch das Hinzufügen des Tupels nicht. Dies bedeutet, daß implizit eine semantische Gleichheit der beiden syntaktisch identischen Tupel angenommen wird, d.h. daß beide Tupel den gleichen Gegenstand repräsentieren.
- Die ω-Relation ändert sich durch das Hinzufügen des Tupels nicht, die Interpretation der Vorkommen von ω in dem betroffenen Tupel lautet aber wie folgt: Jedes derartige Vorkommen von ω repräsentiert eine nichtleere, endliche Menge von definierten Werten. Durch Einfügen von Tupeln ändert sich somit im allgemeinen die Interpretation von ω-Relationen dahingehend, daß ein Tupel mit unbekannten Attributwerten eine nichtleere Menge von Gegenständen statt immer genau einen Gegenstand repräsentiert.
  Anstatt die Interpretation für alle Vorkommen von ω zu ändern, ist auch eine andere Vorgehensweise denkbar: Es werden statt des einen Symbols ω zwei Symbole eingeführt, wovon ein Symbol für genau einen unbekannten Wert steht und das andere Symbol für eine nichtleere Menge unbekannter Werte.
- Die Operation wird abgewiesen, ähnlich wie dies bei Verletzung einer Integritätsbedingung geschieht.

Die erste dieser vier Varianten werden wir nicht genauer betrachten, da wir bei unseren Untersuchungen möglichst nahe am ursprünglichen relationalen Modell bleiben wollen.

Die Anwendung der zweiten Variante ist dann sinnvoll, wenn die Folgerung der semantischen Gleichheit aus der syntaktischen Gleichheit mit dem modellierten Sachverhalt verträglich ist. Dies ist zum Beispiel gewährleistet, wenn das in [Tha89] vorgeschlagene verallgemeinerte Schlüsselkonzept für Relationen mit fehlenden Attributwerten verwendet wird. Dieses Konzept verlangt im Unterschied zum üblichen Schlüsselkonzept nur, daß je zwei Tupel einer Relation in mindestens einer Attributekombination einer fest vorgegebenen Menge solcher Kombinationen definiert und verschieden voneinander sind. Mit dieser Forderung ergibt sich unmittelbar, daß in Relationen erlaubter DB-Zustände jedes Tupel genau einen Gegenstand repräsentiert, auch wenn einige seiner Werte nicht definiert sind.

Die Beschränkung der Vorkommen von ω auf Attribute, die nicht im Primärschlüssel vorkommen, wird zum Beispiel in SQL ([ISO92]) verlangt (in [ISO89] wird dies noch für alle Schlüsselattribute gefordert). Schon in seinen ersten Arbeiten zum relationalen Datenmodell ([Cod70], [Cod72a]) wird von E.F. Codd die Forderung aufgestellt, daß Werte in Attributen des Primärschlüssels stets definiert

sein müssen und daß zu jedem Relationstyp ein Schlüsselkandidat als Primärschlüssel ausgezeichnet wird. Diese Forderung, die später von Codd bekräftigt wird ([Cod79]), stellt eine stärkere Einschränkung der Modellierungsmöglichkeiten dar, als dies auf den ersten Blick erscheinen mag ([Tha89]). Insbesondere in ω-Relationen, durch die Beziehungen zwischen Objekten repräsentiert werden, kann es durchaus vorkommen, daß unbekannte Werte in Schlüsselattributen auftreten, ohne daß eine schlechte Modellierung des Sachverhalts vorliegt.

Beispiel 2.1: Betrachte den Relationstyp **PLANUNG** (PROJEKT, MITARBEITER, STUNDENZAHL). Es sei angenommen, daß ein Projekt mehr als einen Mitarbeiter haben und ein Mitarbeiter in mehr als einem Projekt engagiert sein kann. Zusätzlich soll gelten, daß ein Mitarbeiter für jedes Projekt, dem er zugeteilt ist, eine feste Anzahl Stunden aufwenden soll; der Aufwand einzelner Mitarbeiter kann aber verschieden sein. Unter diesen Annahmen besteht der Schlüssel des Relationstyps aus den beiden Attributen PROJEKT und MITARBEITER. Ist nun zum Beispiel für ein Projekt festgelegt, daß ihm zwei Mitarbeiter mit unterschiedlicher Stundenzahl zugeordnet sind, ist der Name eines dieser Mitarbeiter aber nicht bekannt, dann kann diese Information nicht in einer Relation des Typs **PLANUNG** repräsentiert werden, ohne daß unbekannte Attributwerte im Schlüsselattribut MITARBEITER erlaubt werden.

Beim Ändern und Löschen von Tupeln kann ebenfalls unterschiedlich vorgegangen werden, wobei darauf zu achten ist, daß die Operationsdefinitionen zu der gewählten Einfügeform passen müssen.

- Falls Duplikate zugelassen sind, muß bei den Operationsdefinitionen festgelegt werden, ob alle Duplikate von den Änderungen bzw. der Löschung betroffen sind oder ob die Operation nur auf ein einzelnes Tupel wirken soll.
- Sind unbekannte Attributwerte nur in Nicht-Schlüsselattributen zugelassen, treten keine Probleme auf, wenn die von einer Operation betroffenen Tupel eindeutig bestimmt sind.
- Falls Vorkommen von ω Mengen von definierten Werten repräsentieren und die von einer Operation betroffenen Tupel bekannt sind, wirkt sich diese Änderung automatisch auf die (unbekannte) Menge von Tupeln aus, die durch das Tupel repräsentiert wird. Soll eine Änderung auf eines der repräsentierten Tupel beschränkt bleiben, könnte wie folgt vorgegangen werden: Das zu ändernde Tupel wird kopiert und die Änderungen werden nur in der Kopie vorgenommen. Zu dem "Original" muß dann vermerkt werden, daß es nicht mit Sicherheit in der Relation enthalten ist, da von der durch ein Vorkommen von ω repräsentierten Wertemenge nicht mehr garantiert werden kann, daß sie nicht leer ist.
  Beim Löschen kann ähnlich vorgegangen werden. Da Vorkommen von ω nicht leere Mengen von Werten repräsentieren, wobei solche Mengen auch einelementig sein können, sind mehrere Vorgehensweisen denkbar. Operationen können so definiert werden, daß alle sich qualifizierenden Tupel aus einer Relation entfernt werden, oder es wird das Löschen als Umkehrvorgang zum Einfügen eines einzelnen Tupels gesehen. Bei der zweiten Variante muß der erwähnte Fall beachtet werden, daß bisher keine Duplikate eines Tupels eingefügt wurden. Dieser Fall kann aber am Inhalt einer Relation nicht abgelesen werden. Damit muß er als möglicher Fall behandelt werden, etwa indem eine geeignete Markierung der betroffenen Tupel erfolgt.

Bei der Diskussion dieser Fälle haben wir angenommen, daß die in einer ω-Relation zu ändernden oder zu löschenden Tupel bekannt sind. Werden diese Tupel durch eine Bedingung in einer Operation bestimmt (z.B. bei <update statement search> in SQL) kann der Fall eintreten, daß sich Tupel nicht mit Sicherheit qualifizieren. Damit kann das Ergebnis einer Änderungs- oder Löschoperation nicht eindeutig festgelegt werden. Es muß mit zusätzlichen Informationen zu Tupeln gearbeitet werden, oder es werden - wie in SQL - nur diejenigen Tupel von einer Operation betroffen, für die die Bedingung der Operation mit Sicherheit erfüllt ist. Zusätzliche Information für ein möglicherweise betroffenes Tupel könnte bei Änderungsoperationen sein, daß die Werte in Attributen, für die die Operation Änderungen vornimmt und die definiert sind, nicht mit Sicherheit gültig sind. Für Löschoperationen könnte dem Tupel insgesamt die Information zugeordnet werden, daß es nicht mehr mit Sicherheit in der Relation vorhanden ist.

Die angesprochene Problematik muß auch beachtet werden unter der Annahme, daß unbekannte Attributwerte nur in Nicht-Schlüsselattributen vorkommen. Betrachte dazu folgendes Beispiel: In einer ω-Relation sind zwei Tupel t = (a,b,c) und t' = (a',b,ω) über der Attributmenge {A,B,C} enthalten und die Bedingung einer Operation lautet "B = b und C = c". Während für das Tupel t mit Sicherheit gilt, daß es die Bedingung erfüllt, kann für t' eine gesicherte Aussage bezüglich der Bedingung wegen des unbekannten Wertes in C nicht gemacht werden.

Bemerkung: Um Wertebereichselemente von Konstantensymbolen in Anfragen und von Werten in informellen Beschreibungen zu unterscheiden, verwenden wir durchgehend unterschiedliche Zeichensätze.

Bei den Einfügeoperationen haben wir oben Fälle betrachtet, in denen syntaktische Duplikate in einer betroffenen ω-Relation eine Rolle spielen. Probleme treten auch dadurch auf, daß Operationen Tupel betreffen, die sich zwar in den Attributen, in denen sie beide definierte Werte haben, nicht unterscheiden, die aber nicht identisch sind. Drei Fälle sind hier zu beachten:
Sei R eine ω-Relation, in die ein Tupel t eingefügt werden soll.

- Es gibt in R ein Tupel t', das in allen definierten Werten mit t übereinstimmt und das in mindestens einem Attribut, in dem t definiert ist, einen undefinierten Wert hat.
  Beispiel: t = (a,b,c,ω), t' = (a,b,ω,ω)
- Der Fall, der durch Vertauschung von t und t' im ersten Fall vorliegt.
- Es gibt in R ein Tupel t', so daß t und t' in allen Attributen übereinstimmen, in denen beide Tupel definierte Werte haben.
  Beispiel: t = (a,b,c,ω), t' = (a,b,ω,d)

Unter der Annahme, daß keine unbekannten Werte in Schlüsselattributen vorkommen, sind folgende Vorgehensweisen denkbar: In allen drei Fällen wird die Einfügeoperation abgewiesen, oder es wird stets das "Maximum" an Information in die Datenbank aufgenommen, d.h. in den drei aufgeführten Fallbeispielen wird das Tupel t' im ersten Fall durch (a,b,c,ω) und im dritten Fall durch (a,b,c,d) ersetzt; im zweiten Fall erfolgt keine Änderung der Relation.

Kommen in t oder t' unbekannte Attributwerte in Schlüsselattributen vor, kann im Fall von Tabellen wie bei der Operationsdefinition für totale Relationen vorge-

gangen werden. Falls das Mengenkonzept beibehalten werden soll, kann t eingefügt werden, ohne daß die Interpretation der Vorkommen von ω in einem Tupel geändert werden muß. Der häufig zu findende Vorschlag, im ersten Fall wie oben t' durch t zu ersetzen, ist nicht mit der Annahme verträglich, daß nichts über die Vorkommen von ω in t' und t bekannt ist. Ist bekannt, daß beide Tupel Information zum gleichen Gegenstand repräsentieren, könnte diese Wissen dadurch eingebracht werden, daß zunächst t' gelöscht und dann t eingefügt wird.
Bei Änderungs- und Löschoperationen ergeben sich ähnliche Konsequenzen.

Die Betrachtungen zeigen, daß bei der Festlegung der Bedeutung von update-Operationen auf ω-Relationen die äußere Semantik eine wichtige Rolle spielt. Ohne Beschränkungen bezüglich des Vorkommens unbekannter Attributwerte ist das Konzept der ω-Relation nicht mächtig genug, um bei der Durchführung einer Einfügeoperation immer so "Buch führen" zu können, daß anschließende Änderungs- und Löschoperationen gemäß der erfolgten Einfügungen ausgeführt werden können (vgl. [AG85]). Es ist hier demnach mit einem Informationsverlust zu rechnen. Andererseits erscheint die Beschränkung der Vorkommen unbekannter Attributwerte auf Nicht-Schlüsselattribute als zu restriktiv, da sie für viele DB-Schemata das Speichern nicht vollständig bekannter Beziehungen ausschließt.

Für weitergehende Untersuchungen der update-Problematik sei auf [KW84], [KW85], [AG85], [Win88] und [Gra91] verwiesen.

2.1.2 Einige grundlegende Begriffe und Eigenschaften

DB-Relationen werden häufig formal als Mengen von Funktionen eingeführt, um einfache Schreibweisen zu erhalten. Tupel, als Funktionen betrachtet, bilden in dieser Sichtweise Attribute auf Werte aus den zugehörigen Wertebereichen ab. Beim Übergang zu DB-Relationen mit einem speziellen Symbol für Vorkommen unbekannter Attributwerte unter Beibehaltung der Mengeneigenschaft auf syntaktischem Niveau bietet es sich an, Tupel als partiell definierte Funktionen einzuführen.

Definition: Ein ω-Tupel über einer Attributmenge $\{A_1,....,A_k\}$ mit Wertebereichsfunktion dom ist eine partielle Funktion

$$t \mid \{A_1,....,A_k\} \rightarrow \bigcup_{i=1}^{k} dom(A_i) \text{ mit}$$

$t(A_i) \in dom(A_i)$ oder $t(A_i)$ nicht definiert, $i = 1,....,k$.

Eine ω-Relation über einer Attributmenge α ist eine endliche Menge von ω-Tupeln über α.

Zur Einführung von Zuständen mit ω-Relationen benötigen wir zunächst noch die Definitionen von Relationstypen und relationalen DB-Schemata.

Definition: Sei $\{A_1,....,A_k\}$, $k \geq 1$, eine endliche Menge von Attributen mit einer Wertebereichsfunktion $dom \mid \{A_1,....,A_k\} \rightarrow \{D_1,....,D_s\}$, $s \geq 1$, wobei $D_1,....,D_s$ nichtleere Wertebereiche (domains) sind. Ein Relationstyp über $\{A_1,....,A_k\}$ ist gegeben durch die Attributmenge $\{A_1,....,A_k\}$ zusammen mit einem Bezeichner RT.

Als Schreibweisen für Relationstypen verwenden wir
$(RT, \{A_1,....,A_k\})$, $RT(A_1,....,A_k)$ oder auch (RT,α_i), falls α_i eine bekannte Attributmenge ist.

Ein Relationstyp wird im folgenden häufig mit seinem Bezeichner identifiziert. Die Attributmenge eines Relationstyps RT ist dann durch Attr(RT) gegeben.

Bemerkung: In Beispielen gehen wir zur Vereinfachung stets davon aus, daß die Attribute eines Relationstyps geordnet sind gemäß der Reihenfolge ihres Auftretens bei der Angabe der Attributmenge. Ferner nehmen wir für die "ABC"-Beispiele an, daß alle Attribute die natürlichen Zahlen als Wertebereich haben, wenn nichts anderes gesagt wird.

Definition: Ein relationales Datenbankschema σ ist eine endliche, nichtleere Menge $\{(RT_1,\alpha_1),....,(RT_m,\alpha_m)\}$ von Relationstypen über Teilmengen einer gemeinsamen Attributmenge α mit zugehörigen Wertebereichen $D_1,....,D_s$ und mit einer Wertebereichsfunktion $dom \mid \alpha \rightarrow \{D_1,....,D_s\}$.
Folgende Eigenschaften sollen dabei gelten:

- Alle Bezeichner von Relationstypen sind paarweise verschieden und verschieden von allen Attributen in α,
- die Attribute in α sind verschieden von allen Wertebereichselementen,
- die Wertebereichsfunktion ist surjektiv und
- für die Attributmengen der Relationstypen gilt: $\bigcup_{i=1}^{m} \alpha_i = \alpha$.

Definition: Ein Zustand zu einem DB-Schema $\sigma = \{(RT_1,\alpha_1),....,(RT_m,\alpha_m)\}$ ist eine Abbildung z, die jedem Relationstyp $(RT_i,\alpha_i) \in \sigma$ eine DB-Relation R_i über α_i zuordnet.
Die Menge aller Zustände, die zu einem Schema σ gehören, wird mit $\mathfrak{Z}_\sigma$ bezeichnet. Man kann $\mathfrak{Z}_\sigma$ als modellinhärente Semantik von σ ansehen.

Definition: Ein ω-Zustand zu einem DB-Schema $\sigma = \{(RT_1,\alpha_1),....,(RT_m,\alpha_m)\}$ ist eine Abbildung z_ω, die jedem Relationstyp $(RT_i,\alpha_i) \in \sigma$ eine ω-Relation über α_i, $i = 1,....,m$, zuordnet.

ω-Tupel können bezüglich ihres Informationsgehalts partiell geordnet werden.

Definition: Sei t ein ω-Tupel über einer Attributmenge $\{A_1,....,A_k\}$. Ein ω-Tupel t' über $\{A_1,....,A_k\}$ ist eine Überdeckung von t (in Zeichen: $t' \geq t$ oder $t \leq t'$) $\Leftrightarrow_{df}$
Für alle $A \in \{A_1,....,A_k\}$ mit t(A) ist definiert (d.h. ungleich ω) gilt: t'(A) ist definiert und gleich t(A).
Wir sagen auch: t' überdeckt t oder t wird von t' überdeckt.
Gilt außerdem, daß t' total definiert ist auf $\{A_1,....,A_k\}$, dann heißt t' eine Vervollständigung von t (in Zeichen: $t' \trianglerighteq t$).

Die Menge der Attribute, auf denen ein ω-Tupel t definiert ist, bezeichnen wir mit Def(t).

Bemerkung: Man kann auch zunächst Vervollständigungen zu einem ω-Tupel t definieren als Tupel, die durch Ersetzung aller Vorkommen von ω in t durch beliebige gewöhnliche Werte erhalten werden. Dann läßt sich die Überdeckungsrelation wie folgt definieren: Ein Tupel t' ist eine Überdeckung eines Tupel $t \Leftrightarrow_{df}$
Jede Vervollständigung von t' ist auch eine Vervollständigung von t. Damit wird besonders deutlich, daß die Überdeckung eines ω-Tupels eine genauere Information darstellt als das überdeckte Tupel.

Zu zwei beliebigen ω-Tupeln t, t' über einer Attributmenge α gibt es stets ein bezüglich der Überdeckungsrelation maximales ω-Tupel, das von t und t' überdeckt

wird. Es ist eindeutig bestimmt und kann aus t oder t' wie folgt erhalten werden: Ersetze alle definierten Werte in t (t'), die nicht mit den entsprechenden Werten in t' (t) übereinstimmen, durch ω. Dieses ω-Tupel ist eindeutig bestimmt und soll mit inf(t,t') bezeichnet werden, da es das Infimum für {t,t'} bezüglich der Überdeckungsrelation darstellt.

Beispiel: Seien t = (1,2,ω) und t' = (3,2,1) ω-Tupel über der gleichen Attributmenge. Dann gilt inf(t,t') = (ω,2,ω).

Zu t und t' muß es kein Tupel t'' geben, das sowohl t als auch t' überdeckt (siehe Beispiel). Ein solches Tupel existiert aber immer genau dann, wenn t und t' wie folgt zueinander in Beziehung stehen:

Definition: Seien t und t' ω-Tupel über der gleichen Attributmenge α. t und t' heißen verträglich (in Zeichen: $t \Delta t') \Leftrightarrow_{df} \neg (\exists A \in Def(t) \cap Def(t'))(t(A) \neq t'(A))$.

Sind zwei ω-Tupel t, t' verträglich, dann können sie den gleichen Gegenstand repräsentieren, da sie sich in ihren definierten Werten nicht widersprechen. In diesem Fall gibt es ein bezüglich der Überdeckungsrelation minimales ω-Tupel, das t und t' überdeckt. Es ist eindeutig bestimmt und kann wie folgt aus t oder t' erhalten werden: Ersetze jedes Vorkommen von ω in t (t') durch den Wert von t' (t) in dem entsprechenden Attribut. Wir bezeichnen dieses Element mit sup(t,t'), da es sich um das Suprenum von {t,t'} bezüglich der Überdeckungsrelation handelt.

Beispiel: Seien t = (1,2,ω) und t' = (ω,2,3) ω-Tupel über der gleichen Attributmenge. Dann gilt sup(t,t') = (1,2,3).

Für jede Attributmenge α kann somit {t | t ω-Tupel über α} als Infimum-Halbverband betrachtet werden mit den totalen Tupeln über α als maximalen Elementen und (ω,....,ω) als kleinstem Element. Die maximalen Elemente sind die totalen Tupel über α.

Der Überdeckungsbegriff läßt sich auf ω-Relationen übertragen. Mit seiner Hilfe kann die Bedeutung von ω-Relationen formal festgelegt werden.

Definition: Seien R und S ω-Relationen über der gleichen Attributmenge α. R heißt eine Überdeckung von S (in Zeichen: $R \geq S) \Leftrightarrow_{df}$

$(\forall t \in S)(\exists t' \in R)(t' \geq t)$.

R und S heißen überdeckungsäquivalent (in Zeichen: $R \sim S) \Leftrightarrow_{df}$

$R \geq S$ und $S \geq R$.

Ist R eine Überdeckung von S und total, dann heißt R auch Ausdehnung von S (in Zeichen: $R \trianglerighteq S$).

Kann R aus S erhalten werden, indem einige Vorkommen von ω in S durch einen Wert des zum jeweiligen Vorkommen gehörenden Wertebereichs ersetzt werden, dann sprechen wir von R als einer Erweiterung von S. Eine Erweiterung heißt echt, falls mindestens ein Vorkommen von ω ersetzt wird.

Beispiel 2.2: Betrachte folgende ω-Relationen über {A,B,C}:

R

A	B	C
ω	4	5
ω	4	ω
3	4	ω

S

A	B	C
3	4	5
2	3	ω

T

A	B	C
3	4	ω
ω	ω	5
ω	4	5

Es gilt: $S \geq R$, $S \geq T$ und $R \sim T$.

Die folgende Relation ist eine Erweiterung von R, aber keine Erweiterung von T:

A	B	C
ω	4	5
1	4	ω
3	4	ω

Im folgenden stellen wir einige einfache Eigenschaften zusammen, die für Überdeckungen und Erweiterungen gelten. Seien R und S ω-Relationen über der gleichen Attributmenge.

- Gilt $R \geq S$ und $S \geq R$, d.h. sind R und S überdeckungsäquivalent, dann haben die beiden Relationen die gleichen maximalen Elemente bezüglich $\geq$.

  Beweis: Sei t maximal in R und nicht maximal in S. Wegen $S \geq R$ gibt es dann ein $t' \in S$ mit $t' \neq t$ und $t' \geq t$. Wegen $R \geq S$ gibt es in R ein t'' mit $t'' \geq t'$. Damit kann t nicht maximal sein in R.

- Aus $R \geq S$ und $S \geq R$ folgt im allgemeinen nicht, daß R und S auch die gleichen minimalen Elemente haben. Ein Beleg hierfür sind die beiden Relationen R und T in Beispiel 2.2. Damit kann insbesondere aus der Überdeckungsäquivalenz zweier Relationen nicht auf die Gleichheit der Mengen ihrer Erweiterungen geschlossen werden.

- Ist R eine Erweiterung von S und S eine Erweiterung von R, dann gilt $R = S$.

- Aus der Gleichheit der Menge aller Überdeckungen oder der Menge aller Erweiterungen von R und S folgt die Überdeckungsäquivalenz von R und S (Lemma 12.1 in [Mai83]).

Stimmen für zwei ω-Relationen über der gleichen Attributmenge die Mengen ihrer echten Erweiterungen überein, dann müssen die beiden Relationen nicht nur überdeckungsäquivalent sondern sogar identisch sein, wenn die Wertebereiche der Attribute genügend groß sind.

Lemma 2.1: Seien R und S ω-Relationen über der gleichen Attributmenge α mit Wertebereichsfunktion dom, und seien alle durch dom zugeordneten Wertebereiche genügend groß. Dann folgt aus der Gleichheit der Mengen aller echten Erweiterungen von R und S die Gleichheit von R und S selbst.

Beweis: R und S sind überdeckungsäquivalent. Außerdem haben die beiden Relationen die gleiche Anzahl Elemente: Wähle für jedes Attribut aus α einen Wert, der in beiden Relationen nicht vorkommt, und ersetze alle Vorkommen von ω für jedes Attribut durch den jeweils gewählten Wert. Durch diese Einsetzung ändert sich die Anzahl der Tupel in den Relationen nicht. Da keine Tupel durch die Einsetzung identifiziert werden, haben die erzeugten Erweiterungen R' bzw. S' die gleiche Tupelanzahl wie die ursprünglichen Relationen. Da die Tupelanzahl außerdem jeweils maximal ist für die Menge der Erweiterungen, müssen auch R und S die gleiche Tupelanzahl haben.
Wegen der Gleichheit der Erweiterungsmengen muß gelten: $R' \geq S$ und $S' \geq R$. Da die gewählten Werte weder in R noch in S vorkommen, können R' aus S und S' aus R nur wie R' aus R bzw. S' aus S entstehen. Daraus folgt die Gleichheit von R und S. □

Die Eigenschaft, daß alle Wertebereiche genügend groß sind, wird im Beweis benötigt, damit Elemente gewählt werden können, die in den betrachteten Relationen nicht vorkommen. Ohne diese Eigenschaft können Relationen bei gleichen Mengen echter Erweiterungen verschieden voneinander sein, wie das folgende Beispiel zeigt.

Beispiel 2.3: Seien R und S folgende ω-Relationen über der Attributmenge {A,B} mit Wertebereich {1,2} für beide Attribute:

R

A	B
1	2
1	1
1	ω
2	1

S

A	B
1	2
1	1
ω	1
2	1

Jede echte Erweiterung von R ist offensichtlich auch eine Erweiterung von S und umgekehrt. Die Tupel (1,ω) in R und (ω,1) in S sind insofern redundant, als jede ihrer Überdeckungen schon in R bzw. S enthalten ist.

Werden nur Relationen betrachtet, in denen die im Beispiel angesprochene Redundanz nicht gilt, dann muß an Wertebereiche nur die Bedingung gestellt werden, daß sie mindestens zwei Elemente enthalten, um die Aussage von Lemma 2.1 zu erhalten.

Nach diesen Vorbereitungen können wir nun ω-Relationen und ω-Zuständen auch formal eine Bedeutung zuordnen, indem wir angeben, welche totalen Relationen bzw. Zustände aus ihnen gefolgert werden können. Das heißt, die Semantik partieller Relationen und Zustände wird durch Mengen totaler Relationen bzw. Zustände festgelegt. Elemente dieser Mengen heißen mögliche Relationen bzw. mögliche Zustände.

In Abhängigkeit von den oben schon angesprochenen unterschiedlichen Möglichkeiten zur Interpretation einzelner Vorkommen von ω gibt es verschiedene Formen der Semantikfestlegung, d.h. der Zuordnung möglicher Relationen und möglicher Zustände.

2.1.3 Vervollständigungen

Steht in einer ω-Relation jedes Vorkommen von ω für genau einen unbekannten Wert, dann repräsentiert diese partielle Relation die Menge der totalen Relationen, die als sogenannte Vervollständigungen der ω-Relation angesehen werden können.

Definition: Sei R eine beliebige ω-Relation. Eine Erweiterung R' von R heißt eine Vervollständigung (englisch: completion) von R, falls R' total ist.

Beispiel 2.4: Betrachte folgende ω-Relation S über {A,B}:

S

A	B
3	4
3	ω
ω	4

Vervollständigungen von S sind zum Beispiel:

A	B
3	4

A	B
3	4
3	5
2	4

A	B
3	4
4	4

Dagegen ist die folgende Relation keine Vervollständigung von S:

A	B
3	4
3	5
3	6

Sei mit $POSS_c(R)$ die Menge aller Vervollständigungen einer ω-Relation R bezeichnet (POSS steht für possibility und c für completion). $POSS_c$ kann als Funktion aufgefaßt werden, die jeder Relation R über einer Attributmenge α mit Wertebereichsfunktion dom die Menge aller Vervollständigungen zuordnet. Da die Vervollständigungen mögliche Relationen zu R sind, wird $POSS_c$ als Möglichkeitsfunktion oder auch POSS-Funktion bezeichnet ([Mai83]).

Jede ω-Relation R wird durch $POSS_c(R)$ eindeutig charakterisiert, d.h. Lemma 2.1 läßt sich verschärfen zu (vergl. Übungen 12.4 und 12.6 in [Mai83]):

Lemma 2.2: Seien R und S ω-Relationen über einer gemeinsamen Attributmenge α mit Wertebereichsfunktion dom, und seien alle durch dom zugeordneten Wertebereiche genügend groß. Dann gilt

$$POSS_c(R) = POSS_c(S) \Rightarrow R = S.$$

Beweis: Zum Beweis betrachten wir den Beweisgang zu Lemma 2.1. Die Einsetzung der neuen Werte in R und S bedeutet, daß die beiden erzeugten Relationen R' und S' Vervollständigungen sind. Wegen $POSS_c(R) = POSS_c(S)$ ist damit R' auch eine Vervollständigung von S und S' eine Vervollständigung von R. Mit der gleichen Argumentation wie oben folgt damit die Gleichheit von R und S. □

POSS-Funktionen können in natürlicher Weise auf ω-Zustände ausgedehnt werden:

Definition: Seien $\sigma = \{(RT_1,\alpha_1),....,(RT_m,\alpha_m)\}$ und z_ω ein ω-Zustand zu σ. Dann ist

$$POSS_c(z_\omega) =_{df} \{z \in \mathfrak{Z}_\sigma \mid z(RT_i) \in POSS_c(z_\omega(RT_i)),\ i = 1,....,m\}$$

die Menge aller möglichen Zustände zu z_ω gemäß $POSS_c$.

Wir wollen zur Vereinfachung der Schreibweise gelegentlich POSS-Funktionen auch auf einzelne Tupel anwenden. Für ein ω-Tupel t soll dann $POSS_c(t)$ die Menge aller Vervollständigungen von t bezeichnen.

2.1.4 Enge Ausdehnungen

Betrachten wir nun den Fall, daß Vorkommen von ω mehr als einen Wert repräsentieren können. Mögliche Relationen zu einer gegebenen ω-Relation R erhält man unter dieser Annahme wie folgt: Zu jedem Tupel von R werden ein oder beliebig, aber endlich viele Tupel erzeugt, indem alle Vorkommen von ω durch definierte Werte ersetzt werden. Für jedes Tupel t' einer möglichen Relation zu R existiert damit ein Tupel t in R, aus dem t' auf die beschriebene Weise erzeugbar ist.

Definition: Sei R eine ω-Relation über α. Eine totale Relation R' über α ist eine enge Ausdehnung (englisch: close extension) von R, falls R' eine Ausdehnung von R ist mit folgender Eigenschaft: Für jedes $t' \in R'$ gibt es ein $t \in R$ mit $t' \trianglerighteq t$.

Die Menge aller engen Ausdehnungen von R sei mit $POSS_{ce}(R)$ bezeichnet (ce steht für close extension). $POSS_{ce}(z_\omega)$ sei analog zu $POSS_c(z_\omega)$ definiert.

Beispiel 2.5: Betrachten wir wieder die Relation S aus Beispiel 2.4:

S

A	B
3	4
3	ω
ω	4

Enge Ausdehnungen von S sind zum Beispiel:

A	B
3	4
3	5
4	4

A	B
3	4
3	5
3	6
4	4

A	B
3	4
1	4
2	4
4	4

A	B
3	4

Die Möglichkeit, ein ω-Tupel mehrfach bei der Erzeugung totaler Tupel verwenden zu können, führt dazu, daß überdeckte Tupel bezüglich der Menge aller engen Ausdehnungen keine Bedeutung tragen, wenn sie selbst andere Tupel überdecken: Sie schränken nicht ein, da sie überdeckt werden, und sie sind für die Erzeugung überflüssig, da sie selbst überdecken. Es gilt daher:

Lemma 2.3 ([Bis83]): Sei R^- diejenige Relation, die aus der ω-Relation R entsteht, indem aus R alle Tupel t entfernt werden, für die es in R Tupel t' und t'' mit $t \neq t' \neq t''$ gibt, so daß $t' \geq t \geq t''$. Dann gilt

$$POSS_{ce}(R) = POSS_{ce}(R^-).$$

Falls man nur an $POSS_{ce}(R)$ interessiert ist, genügt es somit, sich auf die bezüglich der Überdeckungsrelation maximalen oder minimalen Elemente zu beschränken. Eine ω-Relation, die nur aus solchen Elementen besteht, heißt auch nicht redundant ([Bis83]). Offensichtlich ist die nicht redundante Teilrelation für jede ω-Relation eindeutig bestimmt.

Aus Lemma 2.3 folgt unmittelbar, daß $POSS_{ce}(R) = POSS_{ce}(S)$ nicht die Gleichheit von R und S impliziert, auch wenn die Wertebereiche als unendlich angenommen werden. Für nicht redundante ω-Relationen gilt dagegen:

Lemma 2.4: Seien R und S nicht redundante ω-Relationen über α mit: Die Wertebereiche der Attribute aus α enthalten genügend viele Elemente. Dann gilt

$$POSS_{ce}(R) = POSS_{ce}(S) \Rightarrow R = S.$$

Beweis: Wir betrachten die maximalen und die minimalen Elemente von R und S getrennt.

Aus $POSS_{ce}(R) = POSS_{ce}(S)$ folgt die Überdeckungsäquivalenz von R und S, da diese Folgerung allgemein für Überdeckungen gilt, wie wir oben festgestellt

haben, und da aus $POSS_{ce}(R) = POSS_{ce}(S)$ die Gleichheit der Menge der Überdeckungen von R und S folgt.

Sei t maximales Element von R. In jedem $R' \in POSS_{ce}(R)$ gibt es nach Definition der engen Ausdehnung dann ein t' mit $t'_{|Def(t)} = t_{|Def(t)}$. Wegen $POSS_{ce}(R) = POSS_{ce}(S)$ und der Voraussetzung, daß alle Wertebereiche genügend viele Elemente enthalten, gibt es damit in S ein Element s mit $s \geq t$. Ist $s \neq t$, dann gibt es in jedem $S' \in POSS_{ce}(S)$ ein s' mit $s'_{|Def(s)} = s_{|Def(s)}$ und $Def(t) \subsetneq Def(s)$. Dies steht aber im Widerspruch dazu, daß t maximal ist in R. Somit gilt s = t, und s ist maximal in S. Da auch jedes maximale Element von S mit der analogen Argumentation maximales Element von R ist, haben beide Relationen die gleichen maximalen Elemente.

Sei t minimales Element von R. Es gibt in S ein Element s mit $t \geq s$, da sonst ein Tupel t', $t' \trianglerighteq t$, in einer Relation $R' \in POSS_{ce}(R)$ vorkommen kann, das für kein Tupel aus S eine Überdeckung darstellt. Hier fließt wiederum die Annahme genügend großer Wertebereiche ein. Ist $s \neq t$, dann gibt es in jedem $S' \in POSS_{ce}(S)$ ein Tupel s' mit $s'_{|Def(s)} = s_{|Def(s)}$ und $Def(s) \subsetneq Def(t)$. Dies steht aber im Widerspruch zur Minimalität von t in R. Somit gilt s = t, und s ist minimal in S. Damit folgt wie oben für die maximalen Elemente auch für die minimalen Elemente die Gleichheit. □

Unter der Voraussetzung, daß die Wertebereiche der Attribute genügend viele Elemente enthalten, sind damit sowohl bei Vervollständigungen als auch bei engen Ausdehnungen (nicht redundante) ω-Relationen durch die zugehörigen Mengen möglicher Relationen eindeutig bestimmt. Dies bedeutet zum Beispiel für Antworten, daß deren Definition über diese Mengen erfolgen kann, ohne daß Information verlorengehen muß.

Bei Berücksichtigung von Integritätsbedingungen sind enge Ausdehnungen stets Vervollständigungen, wenn unbekannte Attributwerte nur in Nicht-Schlüsselattributen vorkommen dürfen und nur enge Ausdehnungen zugelassen werden, die alle die Schlüssel betreffenden Integritätsbedingungen erfüllen. Die Ersetzung eines unbekannten Wertes durch mehr als einen Wert würde in solchen Fällen zur Verletzung der Schlüsseleigenschaft führen.

Die enge Ausdehnung als POSS-Funktion paßt zur *closed world assumption* ([Rei78], kurz CWA), gemäß der für eine Information die Nicht-Gültigkeit angenommen werden darf, wenn sie sich aus der Datenbank nicht ableiten läßt. Die sogenannte *open world assumption* ([Rei78a], kurz OWA), die dies nicht erlaubt, läßt sich aber nachbilden: Es genügt, einer ω-Relation über α das total undefinierte Tupel über α, (ω,....,ω), hinzuzufügen. Dieses Tupel wird von allen totalen Tupeln über α überdeckt.

2.1.5 Mischformen

Es liegt nahe, die Begriffe Vervollständigung und enge Ausdehnung nicht auf Relationen sondern auf einzelne Attribute zu beziehen. Enge Ausdehnungen sind für Nicht-Schlüsselattribute nur von Bedeutung, wenn auch Vorkommen von ω in Schlüsselattributen vorhanden sind. Geben für alle Attribute eines Schlüssels die zu modellierenden Fakten vor, daß jeder unbekannte Attributwert genau einen konkreten Wert repräsentiert, dann ist für sämtliche Attribute nur die Vervollständigung sinnvoll. "Mischformen" möglicher Relationen zu einer partiellen Relation lassen sich auf einfache Weise in zwei Stufen definieren: Zu einer ω-Relation R werden zunächst alle ω-Relationen erzeugt, die Vervollständigungen von R sind, wobei alle Vorkommen von ω in Attributen, für die die enge Ausdehnung angenommen wird, als Vorkommen

einer gewöhnlichen Konstanten des entsprechenden Wertebereichs betrachtet, d.h. nicht ersetzt werden. Anschließend werden zu allen erzeugten Relationen die engen Ausdehnungen gebildet. Die Gesamtheit der auf diese Weise erzeugbaren totalen Relationen bildet die Menge der möglichen Relationen zu R.

2.2 V-Relationen und V-Zustände

Die Modellierung unbekannter Attributwerte durch einen einzigen speziellen Wert hat den Nachteil, daß die "Stellen", an denen dieser Wert vorkommt, für die Auswertung von Anfragen nicht direkt als Information zur Verfügung stehen. Damit ist zum Beispiel bei der Auswertung von Ausdrücken der Relationenalgebra im allgemeinen nicht unmittelbar ablesbar, daß es sich bei zwei Vorkommen von ω in den Operanden einer Operation um "Kopien" des gleichen Vorkommens von ω in einer Relation des betrachteten Zustands handelt. Ein solches Wissen ergibt sich für manche Ausdrücke über die Analyse ihres syntaktischen Aufbaus.

Durch Verwendung eindeutiger Variablen anstelle eines einzigen Symbols kann eine solche Information direkt zur Verfügung gestellt und ausgenutzt werden. Relationen mit Variablen für das Markieren von Stellen mit unbekannten Attributwerten sollen V-Relationen heißen (in Anlehnung an den Begriff v-table aus [IL83]). Wir wollen im folgenden annehmen, daß in V-Relationen eines Zustands jedes Vorkommen eines unbekannten Attributwertes durch eine Variable eindeutig markiert ist. Durch die Forderung der Eindeutigkeit der Markierung wird ausgeschlossen, daß mehr Information in einem Zustand repräsentiert werden kann, als wir mit dem Vorkommen von unbekannten Attributwerten bisher verknüpfen: Das mehrfache Auftreten einer Variablen würde bedeuten, daß es sich an den entsprechenden Stellen immer um den gleichen Wert handelt, der nicht bekannt ist.

Im Unterschied zu ω-Relationen diskutieren wir update-Probleme für V-Relationen erst im Anschluß an ihre formale Einführung.

2.2.1 Einige Begriffe und Eigenschaften

Sei α eine endliche Attributmenge mit Wertebereichsfunktion $dom \mid \alpha \rightarrow \{D_1,....,D_s\}$, und sei $DOM_\alpha = \bigcup_{i=1}^{s} D_i$.

Für jedes $A \in \alpha$ sei V_A eine abzählbar unendliche Menge von Variablen derart, daß $V_A \cap DOM_\alpha = \phi$, $V_A \cap V_B = \phi$, falls $dom(A) \neq dom(B)$, und $V_A = V_B$, falls $dom(A) = dom(B)$. Sei ferner $V = \bigcup_{A \in \alpha} V_A$.

Definition: Sei $\alpha_i \subseteq \alpha$. Ein V-Tupel t über α_i ist eine Abbildung $t \mid \alpha_i \rightarrow DOM_\alpha \cup V$ mit $t(A) \in dom(A) \cup V_A$ für alle $A \in \alpha_i$.
Eine V-Relation R über α_i ist eine endliche Menge von V-Tupeln über α_i.
Ein V-Zustand z_V zu einem DB-Schema $\sigma = \{(RT_1,\alpha_1),....,(RT_m,\alpha_m)\}$ über α ist eine Abbildung $z_V \mid \sigma \rightarrow \{R_1,....,R_m\}$ mit $z_V((RT_i,\alpha_i)) = R_i$, R_i V-Relation über α_i für $i = 1,....,m$, und folgender Eigenschaft: Jede Variable $v \in V$ tritt höchstens einmal in den Relationen von z_V auf.

Da jedes Vorkommen eines unbekannten Attributwertes in einem V-Zustand eindeutig durch eine Variable gekennzeichnet ist, bietet es sich an, die Zuordnung von möglichen Zuständen zu V-Zuständen mit Hilfe einer Belegungsfunktion vorzunehmen.

Definition: Eine Belegungsfunktion für eine gegebene Attributmenge α mit zugehöriger Variablenmenge V (kurz Belegung für α,V) ist eine Funktion $\nu \mid V \rightarrow DOM_\alpha$ mit $\nu(v) \in dom(A)$ für $v \in V_A$.

Zur Vereinfachung der Schreibweise kann der Definitionsbereich von ν auf DOM_α ausgedehnt werden, indem $\nu(c) = c$ für alle $c \in DOM_\alpha$ festgelegt wird. Damit läßt sich jede Belegung in natürlicher Weise fortsetzen auf V-Tupel, V-Relationen und V-Zustände.

Definition: Für ein V-Tupel t über $\alpha_i \subseteq \alpha$ und eine Belegung ν für α,V sei $\nu(t)$ gegeben durch

$\nu(t)(A) =_{df} \nu(t(A))$ für alle $A \in \alpha_i$.

$\nu(t)$ ist eine Vervollständigung von t.
Wir nennen eine Vervollständigung von t auch ein mögliches Tupel zu t.
Für eine V-Relation R über $\alpha_i \subseteq \alpha$ und eine Belegung ν für α,V sei

$\nu(R) =_{df} \{\nu(t) \mid t \in R\}$.

Eine solche totale Relation $\nu(R)$ heißt mögliche Relation zu R.
Die Menge aller möglichen Relationen zu R bezeichnen wir mit $POSS_V(R)$.
Für einen V-Zustand z_V zu einem DB-Schema $\sigma = \{(RT_1,\alpha_1),....,(RT_m,\alpha_m)\}$ über α sei $\nu(z_V)$ gegeben durch

$\nu(z_V)(RT_i) =_{df} \nu(z_V(RT_i))$ für i = 1,....,m.

Beispiel: Betrachte folgende V-Relation R über {A,B,C}:

R

A	B	C
1	ω_1	4
1	2	ω_2
1	ω_3	ω_4

Zwei mögliche Relationen zu R sind:

R_1

A	B	C
1	3	4
1	2	5
1	3	6

R_2

A	B	C
1	2	4

R_1 und R_2 werden aus R mit folgenden Belegungen erhalten:

$\nu(\omega_1) = \nu(\omega_3) = 3$, $\nu(\omega_2) = 5$, $\nu(\omega_4) = 6$ bzw.

$\nu(\omega_1) = \nu(\omega_3) = 2$, $\nu(\omega_2) = \nu(\omega_4) = 4$.

Definition: Seien $\sigma = \{(RT_1,\alpha_1),....,(RT_m,\alpha_m)\}$ ein DB-Schema über α und z_V ein V-Zustand zu σ. Dann ist die Menge aller möglichen Zustände zu z_V (in Zeichen: $POSS_V(z_V)$) wie folgt definiert:

$POSS_V(z_V) =_{df} \{\nu(z_V) \mid \nu$ Belegung für α,V$\}$.

Durch Überdeckungsrelationen sollen Tupel bezüglich ihres Informationsgehaltes geordnet werden. Der Informationsgehalt eines Tupels ergibt sich aus der Menge seiner Vervollständigungen. Wir definieren daher:

Definition: Seien t und t' V-Tupel über der gleichen Attributmenge α. t' ist eine Überdeckung von t (in Zeichen: $t' \geq t$) $\Leftrightarrow_{df}$

$(\forall \nu$: ν Belegung für α,V)$(\exists \nu'$: ν' Belegung für α,V)$(\nu(t') = \nu'(t))$.

Beispiele: Es gilt: $(1,\omega_1,3) \geq (\omega_2,\omega_3,3)$, $(1,1,\omega_1) \geq (\omega_1,1,\omega_2)$;
dagegen gilt nicht $(1,1,\omega_1) \geq (\omega_1,1,\omega_1)$, da zum Beispiel das Tupel $(1,1,2)$ zwar eine Vervollständigung von $(1,1,\omega_1)$, aber keine Vervollständigung von $(\omega_1,1,\omega_1)$ ist.

Belegungsfunktionen ν und ν' in der Definition können Variable, die sowohl in t als auch in t' vorkommen, unterschiedlich belegen. Ist man an einer Überdeckungsrelation interessiert, die zu V-Relationen und V-Zuständen paßt, dann muß berücksichtigt werden, daß Vorkommen gleicher Variablen den gleichen Wert repräsentieren. Dies geschieht in der folgenden Definition:

Definition: Sei t ein V-Tupel über einer Attributmenge $\alpha_i \subseteq \alpha$. Ein V-Tupel t' über α_i ist eine genaue Überdeckung von t (in Zeichen: $t' \geq_g t$) $\Leftrightarrow_{df}$

1) $t' \geq t$

2) $t(A) = t'(A) \vee t'(A) \in DOM_\alpha$ für alle $A \in \alpha_i$

Beispiele: Mit passenden Attributmengen gilt $(1,\omega_1,1) \geq_g (\omega_2,\omega_1,\omega_2)$, $(1,1,1) \geq_g (1,\omega_1,\omega_2)$; dagegen gilt zum Beispiel nicht $(1,\omega_1,1) \geq_g (\omega_1,\omega_1,1)$, weil die erste Bedingung verletzt ist.

Aus der Definition folgt unmittelbar, daß für alle V-Tupel t und t' gilt:
$t' \geq_g t \wedge t \geq_g t' \Leftrightarrow t = t'$.
Ferner läßt sich ebenfalls unmittelbar ablesen, daß jede genaue Überdeckung eines Tupels auch eine Überdeckung ist.

Auch für die Verträglichkeit zweier V-Tupel kann man unterschiedlich strenge Anforderungen stellen. Sei für ein V-Tupel t über α_i mit Def(t) wieder die Menge der Attribute aus α_i bezeichnet, in denen t definiert, d.h. ungleich einem Element aus V ist.

Definition: Seien t und t' V-Tupel über der gleichen Attributmenge $\alpha_i \subseteq \alpha$.
t und t' heißen verträglich (in Zeichen: $t \Delta t'$) $\Leftrightarrow_{df}$

$(\exists \nu$: ν Belegung für α,V)$(\nu(t) = \nu(t'))$.

t und t' heißen stark verträglich (in Zeichen: $t \Delta_s t'$) $\Leftrightarrow_{df}$

$t \Delta t' \wedge (\forall A \in \alpha_i)(A \notin Def(t) \cup Def(t') \Rightarrow t(A) = t'(A))$.

Beispiele: $(1,2,\omega_1) \Delta (\omega_1,2,\omega_2)$, $(1,2,\omega_1) \Delta_s (\omega_1,2,\omega_1)$

Man beachte die Parallelen, die sich zwischen den hier eingeführten Begriffen und den Begriffen *Substitution* und *unifizierbar* bei atomaren Formeln in logischen Datenmodellen ergeben: V-Tupel können als atomare Formeln aufgefaßt werden, wobei ein Prädikatensymbol geeignet zu ergänzen ist (im Zustand zum Beispiel der Bezeichner des entsprechenden Relationstyps). Aus jeder Belegung für eine Attributmenge α und eine Variablenmenge V bestimmt sich zu jedem V-Tupel t über $\alpha_i \subseteq \alpha$ eindeutig eine Grundsubstitution, durch deren Anwendung auf t ein Grundatom erzeugt wird. Zwei V-Tupel sind mit dieser Auffassung verträglich genau dann, wenn sie unifizierbar sind.

Beim Verschmelzen von V-Tupeln oder Teilen von V-Tupeln, was bei der Durchschnittsbildung bzw. bei der Join-Operation notwendig sein kann, ist man im

allgemeinen an möglichst geringem Informationsverlust interessiert. Dies bedeutet z.B. für zwei verträgliche V-Tupel, daß das Ergebnis ihrer Schnittbildung ein bezüglich der Überdeckungsrelation kleinstes Element sein sollte, das Überdeckung von beiden Tupeln ist. Anders ausgedrückt soll die Menge der Überdeckungen des Ergebnisses möglichst groß sein.

Damit können wir zur Definition der Verschmelzung zweier verträglicher V-Tupel auf den Begriff des *allgemeinsten Unifikators* zurückgreifen ([Rob65]). Für Literale gibt es einen einfachen Algorithmus zur Bestimmung eines allgemeinsten Unifikators (siehe etwa [Sch87]). Dieser kann unmittelbar auch für V-Tupel verwendet werden.

Definition: Seien t und t' verträgliche V-Tupel über der gleichen Attributmenge $\alpha_i \subseteq \alpha$. Dann ist eine Verschmelzung von t und t' (in Zeichen: merge(t,t')) ein V-Tupel über α_i, das durch Anwendung eines allgemeinsten Unifikators von t und t' auf eines der beiden Tupel erhalten wird.

Beispiele: merge((1,ω_1,ω_2),(ω_3,2,ω_4)) = (1,2,ω_2) oder (1,2,ω_4)
merge((1,ω_1),(ω_1,ω_1)) = (1,1),
merge((1,2,ω_2),(ω_1,2,ω_1)) = (1,2,1)

Der allgemeinste Unifikator muß nicht eindeutig bestimmt sein; dies zeigt auch eines der Beispiele. Unterschiede können allerdings im Ergebnis nur in den Variablen auftreten und zwar nur dann, wenn schon in den beiden zu verschmelzenden Tupeln in dem entsprechenden Attribut unterschiedliche Variable vorkommen. Die Mengen aller Vervollständigungen sind für alle Verschmelzungen zweier V-Tupel identisch.

Betrachte zwei stark verträgliche V-Tupel t und t'. In diesem Fall entspricht das Verschmelzen von t und t' der Suprenum-Bildung bei ω-Relationen mit anschließender Anpassung aller Variablen an eventuell vorgenommene Ersetzungen durch definierte Werte (siehe zweites Beispiel oben).

Bei der Definition eines Infimums zweier beliebiger V-Tupel entsprechend der Infimum-Definition bei ω-Relationen ergibt sich die Schwierigkeit, daß mit den Variablen mehr als ein Symbol für unbekannte Werte zur Verfügung steht und daß Variable untereinander bezüglich ihres Informationsgehaltes nicht sinnvoll geordnet werden können. Für zwei Tupel (1) und (2) könnte damit jede Variable für ein Infimum hergenommen werden; für (ω_1) und (ω_2) müßte diese Variable ungleich ω_1 und ω_2 sein. Es bietet sich an, neue Variable einzuführen und auf Tupelklassen überzugehen. Wir benötigen dieses Vorgehen bei der Definition von Antworten und führen die entsprechenden Erweiterungen dort ein.

2.2.2 Update-Probleme

Da Variable in V-Relationen nichts weiter als eindeutige Marken für Vorkommen unbekannter Attributwerte sind, muß bei der Definition von update-Operationen darauf geachtet werden, daß sie zu dieser Bedeutung passen. Insbesondere sollte es beim Einfügen von Tupeln daher nicht möglich sein, Variable in den Tupeln zu verwenden. Folgendes Vorgehen bietet sich an:

In einzufügenden Tupeln werden Attribute mit fehlenden Werten einheitlich markiert (oder sie bestimmen sich dadurch, daß Zuweisungen von Werten fehlen). Die Variablen werden wie Konstantensymbole betrachtet; beim Einfügen eines

Tupels werden immer neue Variable für unbekannte Attributwerte generiert; dabei darf eine Variable auch innerhalb eines Tupels nicht mehrfach auftreten.

Das Duplikatproblem, wie es bei ω-Relationen auftritt, stellt sich mit dieser Vorgehensweise nicht. Tupel mit unbekannten Attributwerten, die in allen definierten Werten übereinstimmen, sind durch ihre Variablen unterscheidbar. Der Informationsgehalt einer V-Relation entspricht damit dem einer Tabelle mit ω-Tupeln. In beiden Formen kann die Anzahl von Duplikattupeln, d.h. Tupeln, die in allen definierten Werten übereinstimmen, abgelesen werden.

Duplikattupel können auch ausgeschlossen werden, indem eine Einfügeoperationen solche Tupel abweist bzw. indem sie für Duplikattupel wie die Identität auf einer Relation wirkt.

Beim Ändern und Löschen von Tupeln sind wie bei ω-Relationen zwei Varianten möglich: Entweder wirkt eine Operation auf alle betroffenen Duplikattupel oder nur auf ein Exemplar (das etwa über einen Zeiger bestimmt ist). Probleme mit der zweiten Varianten wie bei ω-Relationen können nicht entstehen.

Beim Einfügen von Tupeln müssen wiederum drei Fälle von Überdeckungen beachtet werden, die auftreten können. Sei R die V-Relation, in die ein Tupel t eingefügt werden soll.

- Es gibt in R ein Tupel t', das in Def(t') mit t übereinstimmt und das zusätzlich in mindestens einem Attribut, in dem t definiert ist, einen undefinierten Wert hat.

  Beispiel: $t = (a,b,c,\omega)$, $t' = (a,b,\omega_1,\omega_2)$

- Der zweite Fall wird durch Vertauschung von t und t' im ersten Fall erhalten.

- Es gibt in R ein Tupel t', das mit t in allen Attributen aus $Def(t) \cap Def(t')$ übereinstimmt.

  Beispiel: $t = (a,b,c,\omega)$, $t' = (a,b,\omega_1,d)$

Sei angenommen, daß keine unbekannten Werte in Primärschlüsselattributen vorkommen. Dann bieten sich zwei Vorgehensweisen an: Die Einfügeoperation wird in allen drei Fällen als nicht erlaubt abgewiesen, oder es wird wie folgt verfahren: Im ersten und dritten Fall werden in t' die Variablen in den Attributen, in denen t definierte Werte hat, durch die entsprechenden Werte aus t ersetzt. Im zweiten Fall erfolgt keine Änderung der Relation.

Treten in t unbekannte Werte in mindestens einem Attribut jedes Schlüssels auf, kann eine Einfügung von t wie im gewöhnlichen Fall erfolgen. Auch hier gilt wie bei ω-Relationen, daß eine Identifizierung von definierten oder unbekannten Werten in t mit Werten eines Tupels aus der Relation, wie sie oben vorgenommen wird, mit den Annahmen bezüglich des Wissens über unbekannte Attributwerte nicht verträglich ist.

Beim Ändern und Löschen treten keine Besonderheiten auf, solange die zu ändernden oder löschenden Tupel eindeutig bestimmt sind. Mit Duplikattupeln kann dabei wie bei ω-Relationen in unterschiedlicher Weise umgegangen werden. Sind die von Operationen betroffenen Tupelmengen nicht eindeutig bestimmt, ergeben sich die gleichen Probleme wie bei ω-Relationen.

Durch das Einführen neuer Variablen bei jeder Einfügung eines Tupels mit unbekannten Attributwerten kann im allgemeinen mehr (differenzierter) Informa-

tion in einer Relation und damit in Zuständen gespeichert werden, als dies bei ω-Relationen und ω-Zuständen möglich ist ([AG85]). Mit den Variablen können Tupel, die in allen definierten Werten übereinstimmen, unterschieden werden, was bei ω-Relationen nicht möglich ist. Dort bleibt nur die Möglichkeit, jedem Tupel genau ein mögliches Tupel (Vervollständigung) oder beliebig viele Tupel zuzuordnen (enge Ausdehnung). Treten in einer V-Relation dagegen k Tupel mit gleichen Werten im definierten Anteil auf, dann bedeutet dies, daß beim Übergang zu einer möglichen Relation höchstens k verschiedene Tupel zu diesen Tupeln "erzeugt" werden. Wie wir aber gesehen haben, kann durch update-Operationen diese Information wieder verlorengehen bzw. genauer, zu unsicherer Information werden.

Bei Beschränkung der Vorkommen von Variablen auf Nicht-Primärschlüsselattribute geht keine Information verloren, wenn in einem V-Zustand alle Variablen durch das Symbol ω ersetzt werden. Es erfolgt mit dieser Ersetzung keine syntaktische Identifizierung von ω-Tupeln, so daß ein zu dem Ausgangszustand äquivalenter V-Zustand wieder erhalten werden kann.

Mit V-Relationen und der gewählten Festlegung möglicher Relationen kann die OWA bei nicht leeren Relationen nicht durch Hinzufügen eines speziellen Tupels "simuliert" werden, wie es bei ω-Relationen mit der engen Ausdehnung als POSS-Funktion möglich ist. Da kein Tupel in einer V-Relation mehrere Tupel in einer zugehörigen Relation repräsentieren kann, ist die Anzahl der Elemente in einer V-Relation stets größer oder gleich der Anzahl der Elemente in einer zugehörigen möglichen Relation.

Sei gefordert, daß in V-Relationen jedes Tupel durch seine definierten Werte eindeutig bestimmt sein muß, wobei die definierten Werte nicht notwendigerweise die Schlüsselwerte enthalten. Dann ergibt sich die gleiche "Duplikatelimination" wie bei ω-Relationen. Als Belegungsfunktion müßte dann eine Funktion gewählt werden, die mögliche Relationen analog den engen Ausdehnungen bei ω-Relationen erzeugt.

Die Betrachtungen zeigen, daß mit V-Relationen die Möglichkeiten, die ω-Relationen bieten, bis auf die OWA nachgebildet werden können. Zusätzlich stehen mit den Variablen eindeutige Bezeichner von Stellen mit unbekannten Attributwerten zur Verfügung, die sich bei der Auswertung von Anfragen ausnutzen lassen. Daher stellt sich natürlich die Frage, warum ω-Relationen überhaupt betrachtet werden sollen. Ein wesentlicher Vorteil von ω-Relationen ist die Einfachheit der Generierung des "Platzhalters" ω. In SQL wird als Platzhalter ein einziger spezieller Wert benutzt, der verschieden ist von jedem definierten Wert und der jedem SQL-Datentyp hinzugefügt wird. Für die Realisierung von V-Relationen muß jedem Datentyp eine genügend große Menge von Werten zugeordnet werden, die verschieden sind von allen Werten des Datentyps. Beim Einfügen eines Tupels in eine Relation muß aus dieser Menge ein Wert ausgewählt werden, der für kein Attribut mit diesem Datentyp als Wertebereich zu diesem Zeitpunkt vergeben ist. Für jeden Datentyp müßte demnach aus Effizienzgründen eine zentrale Vergabe der Variablen erfolgen.

3 Gesicherte und mögliche Antworten

3.1 Interpretation von Antworten

Die Form von Antworten auf Anfragen an relationale Datenbanken wird für die gebräuchlichen Anfragesprachen durch die Wahl der Anfrageformulierung festgelegt. Damit ist die Form einer Antwort in diesen Sprachen unabhängig vom Inhalt der Datenbank, an die sich die Anfrage richtet. Eine Antwort besteht aus einer endlichen Menge oder Multimenge von Tupeln mit Werten, die in der Datenbank gespeichert sind, durch Einsetzung gespeicherter Werte in arithmetische Ausdrücke oder in Funktionsausdrücke der Anfrage erhalten werden oder in der Anfrage als Konstantensymbole vorkommen. Alle Antworttupel sind vom gleichen Typ; dieser Typ ist allein durch die Anfrage festgelegt. Im Fall einer Tupelmenge handelt es sich daher bei einer Antwort um eine DB-Relation (wobei die Spaltennummern die Attribute ersetzen). Im Fall einer Multimenge von Tupeln kann man in Anlehnung an die Sprechweise bei SQL von einer Tabelle sprechen. Damit lassen sich Anfragen formal als Funktionen auffassen, die Zustände in DB-Relationen bzw. Tabellen abbilden. Tabellen wollen wir im folgenden nicht weiter betrachten. Zur Vereinfachung nehmen wir an, daß die Reihenfolge der Werte in Antworttupeln ohne Bedeutung und jede Komponente eines Tupels über ein Attribut eindeutig ansprechbar ist.

Definition: Eine Anfrage q vom Typ β an ein DB-Schema σ ist eine partielle Funktion $q \mid \mathfrak{Z}_\sigma \rightarrow \mathfrak{R}_\beta$, die jedem Zustand $z \in \mathfrak{Z}_\sigma$ von σ eine DB-Relation $R \in \mathfrak{R}_\beta$ zuordnet; dabei ist $\mathfrak{R}_\beta$ die Menge aller DB-Relationen über der Attributmenge β. Durch den Typ einer Anfrage ist damit auch ihr Antworttyp gegeben.

Anfragen an ein DB-Schema σ werden in Anfragesprachen zu σ formuliert. Eine solche Anfragesprache L_σ zu σ ist eine Menge von Ausdrücken zusammen mit einer Bedeutungsfunktion μ, so daß für jeden Ausdruck $e \in L_\sigma$ gilt: $\mu(e)$ ist eine Anfrage an σ. Als Beispiele für Anfragesprachen seien die Relationenalgebra, die Relationenkalküle und SQL genannt. Die Bedeutungsfunktion μ ist für die Relationenalgebra durch die Definitionen der Operationen festgelegt, für die Kalküle durch die jeweilige Interpretationsvorschrift und für SQL durch das im SQL-Report angegebene Regelwerk.

Bemerkung: Die Eigenschaft, daß jede Antwort auf eine Anfrage eine DB-Relation bzw. eine Tabelle ist, bringt den Vorteil der Einfachheit für die Interpretation und erlaubt es, Anfragen zur Definition von Sichten herzunehmen, an die dann wiederum Anfragen in der gleichen Weise wie an einen Zustand gestellt werden können. Andererseits schränkt diese Eigenschaft die Möglichkeiten zur Beschreibung der Information, die zu Anfragen erhalten werden kann, stark ein. Es werden daher in der Literatur auch andere Formen von Antworten betrachtet, z.B. intensionale Antworten, durch die Antworten in flexibler Weise abhängig von der anfallenden Information repräsentiert werden können (für einen Vergleich verschiedener Ansätze siehe [Mot94]). Dabei können für die Darstellung insbesondere Integritätsbedingungen ausgenutzt werden ([Mot89]). Wir werden bei der Auswertung von Anfragen in V-Zuständen kurz auf solche Möglichkeiten eingehen, uns im wesentlichen aber mit der gewöhnlichen, extensionalen Antwortform beschäftigen.

Bei unseren Betrachtungen beschränken wir uns auf die Relationenalgebra, den tupelorientierten Relationenkalkül (kurz TRC) und SQL als Anfragesprachen. In

Anfragen der Relationenalgebra und in TRC-Anfragen soll dabei nicht auf Vorkommen fehlender Attributwerte explizit Bezug genommen werden können. Das heißt, wir betrachten weder Anfragen mit Vergleichsausdrücken der Form $A = \omega$ oder $x.A \geq \omega$ noch ein Prädikat, das die Abfrage nach einem fehlenden Wert in einem Attribut erlaubt.

Vergleichsausdrücke, in denen ein Symbol für einen unbekannten Wert wie ein Konstantensymbol verwendet wird, stellen nur für die Operatoren $=$ und $\neq$ sinnvolle Bedingungen für Anfragen dar. Es erscheint für diese Fälle besser, ein Prädikat wie das IS NULL Prädikat von SQL zu verwenden, um den Unterschied zu gewöhnlichen Wertvergleichen deutlich zu machen. Ein solches Prädikat hat für die folgenden Betrachtungen aber keine Bedeutung, da es stets einen Booleschen Wert liefert. In Kapitel 8 wird das IS NULL Prädikat aber bei einer Methode zur Umformung von SQL-Anfragen Verwendung finden.

Die Relationenalgebra können wir in ihrer ursprünglichen Form, wie sie auch aus Lehrbüchern bekannt ist, als Anfragesprache übernehmen. Beim TRC wollen wir nur Anfragen betrachten, die sogenannte *erlaubte* Anfragen darstellen. Für diese Teilklasse kann die Äquivalenz mit der Relationenalgebra einfacher gezeigt werden, als für Klassen mit eingeschränkteren syntaktischen Bedingungen für Bereichsausdrücke, etwa die von Codd in [Cod72b] betrachtete Variante. Diese Variante muß außerdem - wie im Anhang von [Cod72b] erwähnt - um Projektionen auf Bereichsausdrücken erweitert werden, damit die Äquivalenz mit der Relationenalgebra gilt.

Die folgende Definition erlaubter TRC-Anfragen orientiert sich an der Darstellung in [KK93]. Sie geht zurück auf eine Definition erlaubter Anfragen für den wertebereichsorientierten Kalkül ([TS88]). Für die allgemeine Definition von TRC-Ausdrücken sowie ihre Interpretation bei unbeschränktem oder beschränktem Universum sei auf [KK93] verwiesen.

Erlaubte TRC-Anfragen lassen sich als TRC-Anfragen charakterisieren, für die über syntaktische Einschränkungen gewährleistet ist, daß die Interpretation bei unbeschränktem Universum das gleiche Ergebnis liefert wie die Interpretation bei beschränktem, und damit endlichem, Universum, und für die diese syntaktische Einschränkung sowohl einfach als auch möglichst geringfügig ist.

Wir benötigen zunächst einige Definitionen, durch die das Vorkommen einer Variablen in Teilausdrücken von TRC-Ausdrücken charakterisiert wird. Um im folgenden Wohldefiniertheit zu garantieren, wird für TRC-Ausdrücke verlangt, daß in ihnen jede Variable höchstens einmal durch einen Quantor gebunden ist und daß in ihnen keine Variable sowohl frei als auch gebunden vorkommt.

Definition: Sei $\mathfrak{a}$ ein TRC-Ausdruck zu einem DB-Schema σ mit Attributmenge α. Sei x eine Variable von $\mathfrak{a}$ und A ein Attribut mit $A \in typ(x)$; dabei ist typ(x) eine Teilmenge von α, die jeder Variablen eines TRC-Ausdrucks zu σ zugeordnet ist. Wir definieren rekursiv über den Aufbau von TRC-Ausdrücken, wann das Paar (x,A) positiv (negativ) beschränkt in $\mathfrak{a}$ heißt (kurz: (x,A) positiv (negativ) in $\mathfrak{a}$):

1) (x,A) ist positiv in RT_i x.
2) (x,A) ist positiv in $x.A = c$ und in $c = x.A$.
3) (x,A) ist positiv in $x.A = y.B$ oder in $y.B = x.A$, falls $x.A = y.B$ bzw. $y.B = x.A$ als Term F_i in einer Konjunktion $F_1 \wedge F_2 \wedge \ldots \wedge F_k$ auftritt, in der (y,B) positiv ist.

4) (x,A) ist positiv in $\neg$ F, falls (x,A) negativ ist in F.
5) (x,A) ist positiv in F $\wedge$ G, falls (x,A) positiv ist in F oder in G.
6) (x,A) ist positiv in F $\vee$ G, falls (x,A) positiv ist in F und in G.
7) (x,A) ist positiv in F $\Rightarrow$ G, falls (x,A) negativ ist in F und positiv in G.
8) (x,A) ist positiv in ($\exists$y)(F) oder in ($\forall$y)(F), falls (x,A) positiv ist in F.

Analog ist festgelegt, wann ein Paar (x,A) negativ ist in einem Ausdruck:

1) (x,A) ist negativ in $\neg$ F, falls (x,A) positiv ist in F.
2) (x,A) ist negativ in F $\wedge$ G, falls (x,A) negativ ist in F und in G.
3) (x,A) ist negativ in F $\vee$ G, falls (x,A) negativ ist in F oder in G.
4) (x,A) ist negativ in F $\Rightarrow$ G, falls (x,A) positiv ist in F oder negativ in G.
5) (x,A) ist negativ in ($\exists$y)(F) oder in ($\forall$y)(F), falls (x,A) negativ ist in F.

Definition: Sei α ein TRC-Ausdruck zu einem DB-Schema σ. Eine freie Variable x von α ist positiv (negativ) beschränkt in α, falls für alle A $\in$ typ(x) gilt: (x,A) ist positiv (negativ) beschränkt in α.

Beispiel 3.1: Seien σ = {**LIEFERANT** (LNR, NAME), **EINKAUF** (LNR, ABTNAME, ARTNR)}, typ(x) = typ(s) = {LNR, NAME} und typ(u) = {LNR, ABTNAME, ARTNR}. In den folgenden Ausdrücken gilt für die Variablen:

x und s positiv beschränkt:

LIEFERANT x $\wedge$ LIEFERANT s $\vee$ (LIEFERANT s $\wedge$ x.LNR = s.LNR $\wedge$ x.NAME = s.NAME)
($\exists$x)(LIEFERANT x $\wedge$ ($\exists$s)(LIEFERANT s $\wedge$ x.NAME $\neq$ s.NAME))

u negativ beschränkt:

$\neg$ EINKAUF u
($\forall$u)(EINKAUF u $\Rightarrow$ u.ABTNAME $\neq$ Spielwaren)

x negativ beschränkt und s weder positiv noch negativ beschränkt:

($\forall$x)($\forall$s)((LIEFERANT x $\wedge$ x.NAME = s.NAME) $\Rightarrow$ x.LNR = s.LNR))

x und s weder positiv noch negativ beschränkt:

LIEFERANT x $\vee$ LIEFERANT s

Definition: Ein TRC-Ausdruck α zu einem DB-Schema σ heißt erlaubt, falls folgendes gilt:

1) Jede Variable, die in α frei vorkommt, ist positiv beschränkt in α.
2) Für jeden Teilausdruck ($\exists$x)(F) von α ist die Variable x positiv beschränkt in F.
3) Für jeden Teilausdruck ($\forall$x)(F) von α ist die Variable x negativ beschränkt in F.

Definition: Eine erlaubte TRC-Anfrage an ein DB-Schema σ hat die Form

$(x_1,....,x_\ell)/\ \Phi(x_1,....,x_\ell)$,

wobei $x_1,....,x_\ell$ Variable sind und Φ ein erlaubter TRC-Ausdruck zu σ ist, in dem $x_1,....,x_\ell$ als freie Variable vorkommen. Φ heißt auch Qualifikationsausdruck der Anfrage.
Falls Φ keine freien Variablen hat, schreiben wir kurz / Φ statt ()/ Φ und sprechen von einer ja/nein-Anfrage.

Die Beschränkung auf freie Variable in der Zielliste, d.h. der Verzicht auf Projektionsterme, die gewöhnlich in Ziellisten von TRC-Anfragen zugelassen sind, stellt keine Einschränkung hinsichtlich der Ausdruckskraft dar. Dies ergibt sich

daraus, daß jeder Projektionsterm durch eine Variable ersetzt werden kann, die in Φ geeignet "angebunden" ist.

Beispiel 3.2: Sei σ aus Beispiel 3.1 erweitert um den Relationstyp **ARTIKEL** (ARTNR), und seien typ(x) = {LNR, NAME}, typ(y) = {ARTNR} und typ(u) = typ(w) = {LNR, ABTNAME, ARTNR}.
Erlaubte TRC-Anfragen sind dann zum Beispiel:

(y)/ ARTIKEL y $\wedge$ ($\forall$u)(EINKAUF u $\Rightarrow$ u.ARTNR $\neq$ y.ARTNR)

/ ($\exists$x)($\exists$y)(LIEFERANT x $\wedge$ $\neg$ (ARTIKEL y $\Rightarrow$ ($\forall$w)($\neg$ (EINKAUF w $\wedge$ w.LNR = x.LNR $\wedge$ w.ARTNR = y.ARTNR))))

Nicht erlaubte TRC-Anfragen sind dagegen:

(x)/ $\neg$ LIEFERANT x

(x,y)/ LIEFERANT x $\vee$ ARTIKEL y

/ ($\exists$u)($\forall$y)(ARTIKEL y $\Rightarrow$ (EINKAUF u $\wedge$ u.ARTNR = y.ARTNR))

Bemerkung: In Beispielen machen wir gelegentlich Gebrauch von den folgenden Kurzschreibweisen:

($\forall$x: R x)(φ(x)) für ($\forall$x)(R x $\Rightarrow$ φ(x)) und
($\exists$x: R x)(φ(x)) für ($\exists$x)(R x $\wedge$ φ(x)).

Ist β eine Attributmenge $\{B_1,....,B_k\}$, dann schreiben wir auch:

$x.\beta = y.\beta$ für $x.B_1 = y.B_1 \wedge .... \wedge x.B_k = y.B_k$.

Außerdem verwenden wir wie üblich Projektionsterme in Ziellisten.

Die Interpretation einer Antwort auf eine erlaubte TRC-Anfrage in einem total definierten DB-Zustand erfolgt in gleicher Weise wie die Interpretation einer Relation eines DB-Zustands. Im Hinblick auf die CWA bedeutet dies: Aus dem Fehlen eines Tupels in der Antwort auf eine Anfrage in einem Zustand z kann geschlossen werden, daß die Information, die durch dieses Tupel zusammen mit der Anfrage repräsentiert wird, für den durch z modellierten Sachverhalt nicht gültig ist.

Werden in den Zuständen zu einem DB-Schema partiell definierte Tupel in Relationen zugelassen, muß dies nicht nur beim Auswerten von Anfragen beachtet werden, sondern es ist auch die Form und Bedeutung von Antworten auf Anfragen neu festzulegen. Die CWA ist hierfür "ein wenig zu öffnen" ([Bis81], [Min82]), um die Unvollständigkeit der in Relationen modellierten Information zu berücksichtigen: Sei R eine partielle Relation über einer Attributmenge β, die als Antwort auf eine Anfrage erhalten wird. Auf die Nicht-Gültigkeit der durch ein Tupel t über β repräsentierten Information kann nur dann geschlossen werden, wenn es in R kein Tupel gibt, für das t eine Überdeckung darstellt. Dies gilt sowohl für die ω-Modellierung als auch für die V-Modellierung oder Mischformen dieser Modellierungsarten, da wir die Unabhängigkeit aller Vorkommen von ω bzw. von Variablen in DB-Zuständen angenommen haben. Gibt es Abhängigkeiten zwischen Vorkommen von ω oder Variablen, kann der Schluß auf die Nicht-Gültigkeit dagegen für einige Überdeckungen zulässig sein.

Im Fall von ω-Relationen als Antworten ist die CWA in der angegebenen Weise abzuschwächen und zwar unabhängig davon, welche POSS-Funktion (die enge Ausdehnung oder die Vervollständigung) gewählt wird. Ein Unterschied tritt hier allerdings auf, wenn Implikationen für Antworten betrachtet werden, die komplexer sind als der Schluß auf die Nicht-Gültigkeit der Information, die ein einzelnes

Tupel repräsentiert. Da bei einer Vervollständigung einer ω-Relation R ein partielles Tupel durch genau ein totales Tupel ersetzt wird, kann etwa bei Verwendung dieser POSS-Funktion geschlossen werden, daß die durch $t_1 \wedge t_2$, $t_1 \neq t_2$, repräsentierte Information nicht gültig ist, wenn es in R keine zwei verschiedenen Tupel t'_1, t'_2 gibt, so daß $t_1 \trianglerighteq t'_1$ und $t_2 \trianglerighteq t'_2$. Derartige Information ist auch bei der Auswertung von Anfragen zu berücksichtigen.

Anfragen an totale Zustände haben wir oben als partielle Funktionen definiert, da arithmetische Ausdrücke oder Funktionsausdrücke in Abhängigkeit von eingesetzten Werten undefiniert sein können. Für Sprachen ohne diese Ausdrucksmöglichkeiten, wie z.B. die Relationenalgebra oder die Relationenkalküle in ihrer ursprünglichen Form, sind Anfragen stets total definiert. Wir betrachten im folgenden zur Vereinfachung keine Anfragen mit arithmetischen Ausdrücken oder Funktionsausdrücken. Daher können Anfragen an totale Zustände immer als total definierte Funktionen angesehen werden.

Das Vorkommen unbekannter Attributwerte in DB-Zuständen hat zur Folge, daß auch Vergleichsausdrücken in Anfrageformulierungen einer kalkülorientierten Sprache wie etwa dem TRC nicht immer ein Boolescher Wert zugeordnet werden kann; deshalb ist der Test von Tupeln auf Gleichheit im allgemeinen nicht mehr über den Vergleich der Werte der Tupel in sich entsprechenden Attributen definierbar. Im Unterschied etwa zu einer Division durch Null ist es aber sinnvoll, den Vergleich eines konkreten Wertes eines Wertebereichs mit dem speziellen Wert *unbekannt* (repräsentiert durch ω oder ein Element einer Variablenmenge) oder den Vergleich zweier Vorkommen von *unbekannt* miteinander differenzierter zu betrachten. Wird zum Beispiel ein Vorkommen von *unbekannt* aus einem DB-Zustand mit sich selbst verglichen, kann einem solchen Vergleich in Abhängigkeit vom Vergleichsoperator *wahr* oder *falsch* als Wert zugeordnet werden.

Allgemein läßt sich jedem Booleschen Ausdruck über Vergleichsausdrücken *wahr* oder *falsch* zuordnen, wenn der Wert des Ausdrucks unabhängig davon ist, welche Werte für die Vorkommen von *unbekannt* in einem solchen Ausdruck eingesetzt werden, der Ausdruck also bezüglich dieser Vorkommen eine Tautologie (oder einen Widerspruch) darstellt. Neben dieser einfachen Form von Tautologie können auch komplexe TRC-Ausdrücke mit Quantoren bezüglich vorkommender unbekannter Attributwerte Tautologien darstellen. Dabei kann die Tatsache, daß es sich um eine Tautologie oder einen Widerspruch handelt, vom Zustand abhängen, in dem der Ausdruck ausgewertet wird. Betrachten wir hierzu einige Beispiele zu den in Kapitel 2 definierten Modellierungsformen für unbekannte Werte; als Anfragesprache verwenden wir den TRC. Die Beispielrelation und die Anfragen sind [KK93] entnommen.

Es sei folgende DB-Relation mit selbsterklärender Semantik gegeben:

WETTERDATEN

DATUM	ORT	TMAX	TMIN	NSCHLAG	SONNE
3.11.90	Kiel	15	6	0	5
4.11.90	Hamburg	16	4	ω	2
4,11.90	Dortmund	ω	2	1	3

Betrachte folgende Formulierung einer Anfrage im TRC:

$(x)/$ WETTERDATEN $x \wedge x.NSCHLAG \geq 0$

Unter der Annahme, daß der Wertebereich von NSCHLAG keine negativen Zahlen enthält, ist der Vergleichsausdruck $x.NSCHLAG \geq 0$ für alle einsetzbaren Werte *wahr*. Es ist daher sinnvoll, in die Antwort auf die Anfrage alle Tupel der Relation aufzunehmen.

In der nachfolgenden Anfrageformulierung ergibt die Auswertung der Disjunktion für alle einsetzbaren Werte stets *wahr*. Damit ist das Ergebnis der Anfrage als unabhängig von den Werten der Tupel im Attribut TMAX zu betrachten, wenn die Existenz eines entsprechenden Wertes für jedes Tupel bekannt ist. Mit der Bedeutung, die *unbekannt* unterliegt, ist es daher sinnvoll, auch für diesen speziellen Wert die Disjunktion als Tautologie anzusehen.

$(x.DATUM)/$ WETTERDATEN $x \wedge x.ORT = \text{Dortmund} \wedge (x.TMAX \geq 10 \vee x.TMAX < 10)$

Diese Tautologie ist zustandsunabhängig, d.h. sie kann allein durch Betrachtung der Anfrageformulierung festgestellt werden. Die folgende Anfrageformulierung zeigt dagegen ein Beispiel für eine zustandsabhängige Tautologie.

$(x.DATUM, x.ORT)/$ WETTERDATEN $x \wedge ((x.TMAX > 15 \wedge x.NSCHLAG = 0) \vee (x.SONNE < 3 \wedge x.NSCHLAG > 0))$

Gilt für ein Tupel t sowohl $t(TMAX) > 15$ als auch $t(SONNE) < 3$, dann ist der Wert von t in NSCHLAG für die Qualifikation ohne Bedeutung. Auch im Fall, daß der Wert von t in NSCHLAG unbekannt ist, sollte sich t daher für die Antwort qualifizieren.

Teilausdrücke stellen zustandsunabhängige Tautologien oder Widersprüche dar, wenn sie für alle Zustände des betrachteten DB-Schemas den gleichen Wert haben. Da die Zustandsmenge eines DB-Schemas im allgemeinen durch Integritätsbedingungen eingeschränkt ist, können sich auch zustandsunabhängige Tautologien oder Widersprüche ergeben, die nicht allein durch Betrachtung der Anfragen feststellbar sind. Es sind vielmehr die angegebenen Integritätsbedingungen mit zu berücksichtigen. Die folgende Anfrageformulierung zeigt ein Beispiel.

$(x)/$ WETTERDATEN $x \wedge \neg (\exists y:$ WETTERDATEN $y)(y.ORT = x.ORT \wedge y.DATUM = x.DATUM \wedge y.NSCHLAG \neq x.NSCHLAG)$

Unter der Annahme, daß {ORT, DATUM} ein Schlüssel für den Relationstyp WETTERDATEN ist, stellt der negierte quantifizierte Teilausdruck für alle Relationen, die diese Integritätseigenschaft erfüllen, eine Tautologie dar.

Bemerkung: Es sei an dieser Stelle auf einen Unterschied hingewiesen, der sich für die beiden POSS-Funktionen $POSS_c$ und $POSS_{ce}$ ergibt. Im Fall der Vervollständigung kann für das zweite Tupel der angegebenen Relation WETTERDATEN auch ohne die Berücksichtigung der Schlüsseleigenschaft von {ORT, DATUM} auf die Gültigkeit des negierten Teilausdrucks geschlossen werden. Dies liegt daran, daß jedes Vorkommen von ω für genau einen unbekannten Wert steht. Im Fall der

engen Ausdehnung kann ein entsprechender Schluß nicht gezogen werden, da jedes Vorkommen von ω einen oder mehrere unbekannte Werte repräsentiert.

Die Unabhängigkeit des Wertes eines Qualifikationsausdrucks einer Anfrage von der Ersetzung unbekannter Attributwerte in einem DB-Zustand durch definierte Werte bedeutet, daß der Ausdruck für jeden möglichen Zustand zu dem betrachteten partiellen Zustand den gleichen Wert hat. Dieser Wert wird daher insbesondere für denjenigen möglichen Zustand erhalten, der zur (unbekannten) "Realität" paßt. Die Information, die sich aus der Belegung der freien Variablen für die Antwort ergibt, kann deshalb als *gesichert* bezüglich der Anfrage angesehen werden. Zur Darstellung gesicherter Information können unbekannte Attributwerte notwendig sein, wie auch einige der Beispiele zeigen.

Gesicherte Information

Wir sprechen allgemein von *gesicherter Information* zu einer Anfrage an einen partiellen DB-Zustand, wenn die Information in allen möglichen Zuständen bezüglich der Anfrage gültig ist.

Unter der CWA stellen für totale DB-Zustände alle Tupel des Typs, den die Antworttupel haben, gesicherte Information dar: Sie sind entweder mit Sicherheit in der Antwort enthalten oder mit Sicherheit nicht in der Antwort enthalten. Im Fall partieller DB-Zustände gibt es dagegen neben der gesicherten Information auch noch Information, die nur möglicherweise gültig ist.

Möglicherweise gültige Information

Wir sprechen allgemein von *möglicherweise gültiger Information* zu einer Anfrage an einen partiellen DB-Zustand, wenn die Information in einem oder mehreren, aber nicht in allen möglichen Zuständen bezüglich der Anfrage gültig ist.

Für die von uns betrachteten Formen von Anfragen und Antworten handelt es sich bei möglicherweise gültiger Information zu einer Anfrage um alle Tupel vom Typ der Anfrage, die zwar für einen oder mehrere, nicht aber für alle möglichen Zustände in der Antwort enthalten sind.

Die CWA darf bei partiellen DB-Zuständen nicht in gleicher Weise wie bei totalen DB-Zuständen auf eine Antwortrelation angewandt werden, um aus der darin repräsentierten gesicherten *positiven* Information auf gesicherte *negative* Information (es gilt etwas mit Sicherheit nicht) zu schließen. Gesicherte negative Information muß im allgemeinen explizit repräsentiert werden, wenn exakte Kenntnis darüber gewünscht wird.

Drei Formen von Information in Antworten sind demnach bei partiellen DB-Zuständen zu unterscheiden:

- gesicherte positive Information (es gilt etwas mit Sicherheit),
- gesicherte negative Information (es gilt etwas mit Sicherheit nicht) und
- möglicherweise gültige oder nichtgültige Information.

Da jede möglicherweise gültige Information auch möglicherweise nicht gültig ist und umgekehrt, sprechen wir bei der letzten Form von Information auch kurz nur von möglicherweise gültiger Information.

Bezüglich der Darstellbarkeit der verschiedenen Formen gilt natürlich das gleiche wie bei totalen DB-Zuständen: Durch die Beschränkung auf eine Teilklasse erlaubter Anfragen, die in allen konkreten Anfragesprachen erfolgt, wird die positive

Information mit dem betrachteten Zustand und den Konstanten in der Anfrage beschränkt; das vollständige Aufführen gesicherter negativer Information in Antworten ist dagegen im allgemeinen nicht praktikabel. Ihr teilweises Mitführen bei der Auswertung von Anfragen kann aber zu genaueren Antworten führen, ohne die Effizienz der Auswertung stark zu beeinträchtigen.

Bei möglicherweise gültiger Information ist ein weiterer wichtiger Aspekt zu beachten: Falls diese Information in einer einzigen Antwort zusammengefaßt wird, müssen alle Teilinformationen (Tupel) als unabhängig voneinander betrachtet werden. Das heißt, in einer solchen Antwort geht die Information verloren, welche Kombinationen von Teilinformationen in möglichen Antworten erlaubt sind. Wir gehen hierauf in Abschnitt 3.3 (Mögliche Antworten) näher ein.

Gesicherte und möglicherweise gültige Information zu Anfragen haben wir unter Bezugnahme auf einzelne Antworten in möglichen DB-Zuständen charakterisiert. Im Sinne der oben erwähnten intensionalen Antwortformen kann man auch erweiterte Formen gesicherter oder möglicherweise gültiger Information betrachten, die sich auf die Gesamtmenge der möglichen Antworten bezieht. Als Beispiel betrachte die folgende Anfrage, die sich an die WETTERDATEN-Relation von oben richtet:

$$(x)/\ \text{WETTERDATEN}\ x \wedge x.\text{TMAX} > 14 \wedge x.\text{TMIN} \leq 4$$

Für diese Anfrage ist mit der angegebenen ω-Relation die gesicherte Information verbunden, daß es keine weiteren Orte außer Hamburg gibt, die die Bedingung der Anfrage erfüllen und an denen an irgendeinem Tag Maximal- und Minimalwerte der Temperatur gemessen wurden, die beide höher als in Hamburg lagen.

Ein weiteres Beispiel mag die Möglichkeiten andeuten, die sich durch derartige Information zur Präzisierung möglicherweise gültiger Information eröffnen.

$$(x)/\ \text{WETTERDATEN}\ x \wedge (\exists y{:}\ \text{WETTERDATEN}\ y)(y \neq x \wedge y.\text{TMAX} = x.\text{TMAX})$$

Sowohl in Kiel als auch in Hamburg kann das gleiche Temperaturmaximum wie in Dortmund gemessen worden sein. Alle drei Tupel stellen demnach möglicherweise gültige Information zur Anfrage dar. Mit Sicherheit kann aber ausgeschlossen werden, daß in Kiel und in Hamburg das gleiche Maximum wie in Dortmund gemessen wurde.

Für Auswertungsmethoden ergibt sich damit die interessante Fragestellung, ob derartige Information ohne wesentliche zusätzliche Kosten "protokolliert" und der extensionalen Antwort auf eine Anfrage hinzugefügt werden kann. Insbesondere Aussagen bezüglich der maximalen und minimalen Anzahl von Tupeln in den Antworten für die möglichen DB-Zustände erscheinen hierfür besonders geeignet zu sein.

3.2 Gesicherte Antworten

Unter gesicherten Antworten wollen wir Relationen verstehen, die nur Tupel enthalten, die gesicherte positive Information bezüglich einer Anfrage und des betrachteten Zustands enthalten. Für totale Zustände ist demnach eine Antwort immer eine gesicherte Antwort.

Die folgenden Betrachtungen können für die eingeführten POSS-Funktionen POSS_c, POSS_{ce} und POSS_V zunächst in gleicher Weise erfolgen. Wir schreiben daher einfach POSS ohne Index c, ce oder V und sprechen allgemein von einem partiellen Zustand z_p (außer in Beispielen).

Eine Form gesicherter Antworten, die in der Literatur häufig betrachtet wird, ist die gesicherte totale Antwort ([IL84], [Lip84], [AG85], [Gra91]). Sie schränkt gesicherte Antworten auf totale DB-Relationen ein. Diese Antwortform geht zurück auf eine Definition für Antworten, die im Zusammenhang mit einem speziellen mathematischen Modell von Datenbanken, dem *Informationssystem*, als beste untere Schranke ableitbarer Information eingeführt wurde ([Lip79], [Lip81]). Dort wird eine Datenbank als eine Menge von Objekten mit Attributen betrachtet, wobei die Werte zu den Attributen nicht eindeutig bestimmt sein müssen. In Antworten auf Anfragen treten ebenfalls nur Objekte auf. Diese Sichtweise entspricht der Beschränkung relationaler DB-Schemata auf einen einzigen Relationstyp.

Definition: Seien q eine Anfrage an ein DB-Schema σ, z_p ein partieller Zustand zu σ und POSS eine Möglichkeitsfunktion zu σ. Die gesicherte totale Antwort auf q im Zustand z_p (in Zeichen: $GA_{total}(q,z_p)$) ist definiert als:

$$GA_{total}(q,z_p) =_{df} \bigcap_{z \in POSS(z_p)} q(z).$$

Jedes q(z) in der Definition stellt eine mögliche Antwort dar.

Eine gesicherte totale Antwort enthält genau diejenigen Tupel, die in jeder möglichen Antwort enthalten sind. Diese Antwortform scheint daher auf den ersten Blick eine sinnvolle Möglichkeit zur Definition gesicherter Information in Antworten zu sein. Eine genauere Betrachtung zeigt aber schnell, daß sie im Zusammenhang mit konkreten Anfragesprachen wenig hilfreich ist: Als gesicherte totale Antwort auf eine Anfrage nach dem Inhalt einer Relation des Zustands erhält man nur diejenigen Tupel der Relation, die total definiert sind. Ohne Einschränkung von Antworten auf Relationstypen mit Attributen ohne unbekannte Werte, etwa Schlüsselattribute, wenn diese in Zuständen immer konkrete Werte enthalten müssen, ist diese Antwortform zu restriktiv.

Beispiel 3.3: In einem ω-Zustand z_ω sei folgende Relation enthalten:

FIRMA

NAME	STADT	STRASSE
Party-Service	Freudenstadt	ω
Geheimdienst	ω	ω
Alles	Freudenstadt	Hauptstraße

In der gesicherten totalen Antwort für den Algebraausdruck FIRMA ist nur das Tupel (Alles, Freudenstadt , Hauptstraße) enthalten. Projizieren wir mit FIRMA[NAME] auf das Attribut NAME, erhalten wir dagegen für jedes Tupel der Relation ein Tupel in der gesicherten totalen Antwort. Das *Funktionalitätsprinzip* der Relationenalgebra (f(f') = f∘f' für alle Operationen f,f') gilt also für diese Antwortform nicht.

Beispiel 3.4: Sei ein ω-Zustand z_ω mit den beiden folgenden ω-Relationen über {A,B,C} gegeben:

P

A	B	C
1	2	ω
3	4	5

Q

A	B	C
1	2	ω
3	4	ω

Es gilt:

$$\mathcal{GA}_{total}(\text{"P} \cap \text{Q"}, z_\omega) = \Phi_{\{A,B,C\}}$$

$$\mathcal{GA}_{total}(\text{"P} \cup \text{Q"}, z_\omega) = \{(3,4,5)\}$$

$$\mathcal{GA}_{total}(\text{"P} \setminus \text{Q"}, z_\omega) = \Phi_{\{A,B,C\}}.$$

Dabei ist $\Phi_{\{A,B,C\}}$ die leere DB-Relation über {A,B,C}.

Die Probleme mit der gesicherten totalen Antwort machen deutlich, daß bei der Festlegung einer geeigneten Antwortform unvollständige Information in Antworten zugelassen werden muß. Wie sich zeigen wird, gibt es mehrere sinnvolle Möglichkeiten für die Darstellung unvollständiger Information in Antworten und unterschiedliche Festlegungen für ihre Interpretation. Auswertungsverfahren müssen dazu passen.

Betrachten wir zunächst allgemein, wie gesicherte Information zu Anfragen mit Tupelmengen als Antworten dargestellt werden kann. Wir haben gesicherte Information zu Anfragen als Information charakterisiert, die in allen möglichen Antworten gültig ist. Solche Information kann zum Beispiel sein, daß jede mögliche Antwort zu einer Anfrage nicht leer ist, d.h. ein Tupel enthält. Als Darstellung dieser Information könnte ein Tupel vom Antworttyp dienen, das in allen Attributen einen unbekannten Wert hat; die Vorkommen der unbekannten Werte in diesem Tupel sind dabei als unabhängig voneinander zu sehen. Gibt es einen Attributwert, so daß in jeder möglichen Antwort ein Tupel vorkommt, das diesen Wert im entsprechenden Attribut hat, dann ist ein Tupel mit diesem Attributwert und unbekanntem Wert in allen anderen Attributen offensichtlich eine genauere Information als das Tupel mit unbekanntem Wert in allen Attributen. Überdeckungsbegriffe scheinen daher geeignet, um die Güte gesicherter Information in Antworten relativ zu bewerten.

Es kann sinnvoll sein, für gesicherte Antworten eine andere Modellierungsform und/oder eine andere POSS-Funktion zu wählen als für den partiellen DB-Zustand. Im folgenden Beispiel sei die Vervollständigung als POSS-Funktion für den ω-Zustand gewählt.

Beispiel 3.5: Seien zwei ω-Relationen R und S wie folgt gegeben:

R

A	B
1	1
1	2
1	3

S

A	B
1	ω

Betrachte die Differenz R \ S als Anfrage. Da ω für genau einen unbekannten Wert steht, ist gesicherte Information, daß mindestens zwei Tupel in der Antwort enthalten sind, und daß beide Tupel im ersten Attribut den Wert "1" haben. Für die gesicherte Antwort {(1,ω)} ist als POSS-Funktion die enge Ausdehnung besser geeignet als die Vervollständigung, obwohl auch mit ihr Information verlorengeht: Nicht alle engen Ausdehnungen zu {(1,ω)} sind auch mögliche Antworten. Mit der Vervollständigung werden nur mögliche Relationen erhalten, die keine möglichen Antworten darstellen, sondern höchstens Teilmengen davon.

Auch mit der engen Ausdehnung als POSS-Funktion für einen Zustand kann es gesicherte Information sein, daß für Vorkommen unbekannter Werte in Antworten nur bestimmte Werte in Frage kommen.

Beispiel 3.6: Sei ein DB-Schema $\sigma = \{(R, \{A,B\}), (S, \{A,B\})\}$ mit $POSS_{ce}$ als POSS-Funktion für die Zustände gegeben.
Betrachte die Auswertung der Algebraanfrage

$$(S[A=1]\backslash R) \cup (S[A=2]\backslash R) \cup ((R \bowtie R[A \rightarrow \overline{A}])[A=1 \wedge \overline{A}=2])[A,B]$$

in folgendem ω-Zustand zu σ:

R

A	B
ω	3

S

A	B
1	3
2	3

In diesem Fall sind die möglichen Antworten stets von der Form {(1,3)} oder {(2,3)} oder {(1,3),(2,3)}. Mit {(ω,3)} als gesicherter Antwort geht sowohl mit der Vervollständigung als auch mit der engen Ausdehnung als POSS-Funktion Information verloren. Es bietet sich in diesem Fall an, explizit anzugeben, daß nur die Werte 2 und/oder 3 eingesetzt werden können.

Für die Modellierung unbekannter Attributwerte mit V-Relationen ist es naheliegend, auch in Antworten Variable zu verwenden, um Wissen über Vorkommen unbekannter Werte in bestimmten Tupeln und Relationen eines Zustands repräsentieren zu können.

Beispiel 3.7: Gegeben seien die beiden folgenden V-Relationen R und S:

R

A	B	C
2	ω_1	1
2	ω_2	2

S

A	B
2	1

Für den Ausdruck

$$((R \bowtie R[B \rightarrow \overline{B},\ C \rightarrow \overline{C}])[B \geq \overline{B} \wedge C \neq \overline{C}])[A,B]$$

der Relationenalgebra gilt, daß für jede Belegung ν die Antwort entweder $\{(2,\nu(\omega_1))\}$ oder $\{(2,\nu(\omega_2))\}$ ist. Als V-Relation läßt sich diese Information nur durch Einführung einer Variablen modellieren, die im Zustand, d.h. hier in R und S, nicht vorkommt. Wird stattdessen ω_1 oder ω_2 in der Antwort verwendet, repräsentiert die Antwort eine nicht gesicherte Information, falls für Antworten die gleiche Belegung gilt wie für Zustände. Diese gleiche Interpretation von Variablen in Zuständen und Antworten ist offensichtlich sinnvoll. Im vorliegenden Fall besser geeignet ist die Antwort $\{(2,\omega_1 \vee \omega_2)\}$, d.h. die Erweiterung der Wertebereiche von Attributen um geeignete Ausdrücke mit entsprechend angepaßter POSS-Funktion.

Ein allgemeiner Vergleich des Informationsgehalts, den verschiedene Antwortformen bieten, unabhängig von der Form der Modellierung unbekannter Werte in Zuständen, erscheint nicht sinnvoll. Wegen der unterschiedlichen Überdeckungsbegriffe betrachten wir Formen gesicherter Antworten für die ω- und die V-Modellierung in den folgenden Abschnitten getrennt.

3.2.1 Formen gesicherter Antworten für Anfragen an ω-Zustände

Die Festlegung, welche Tupel des Antworttyps gesicherte Information repräsentieren, kann unabhängig von der Wahl der POSS-Funktion (Vervollständigung oder

enge Ausdehnung) für den DB-Zustand und die Interpretation der Antwort erfolgen. Sei daher angenommen, daß für Antworten eine Möglichkeitsfunktion gegeben ist, die nicht notwendigerweise identisch ist mit der Möglichkeitsfunktion für den Zustand. Zur Unterscheidung wollen wir im folgenden, falls notwendig, die Funktion für Antworten mit A-POSS und diejenige für den Zustand mit Z-POSS bezeichnen. Als gesicherte Information wollen wir diejenigen Tupel des Antwortyps einer Anfrage ansehen, zu denen in jeder möglichen Antwort eine Überdeckung gefunden werden kann. Diese "passen" demnach zu mindestens einem Tupel in jeder möglichen Antwort.

Definition: Sei q eine Anfrage vom Typ β an ein DB-Schema σ im ω-Zustand z_ω. Seien Z-POSS die Möglichkeitsfunktion für z_ω und A-POSS die Möglichkeitsfunktion für die Antwortrelation. Die Menge aller gesicherten Antworttupel zu q im Zustand z_ω (in Zeichen: $\mathcal{GT}(q,z_\omega)$) ist definiert als

$$\mathcal{GT}(q,z_\omega) =_{df} \{t \mid t\ \omega\text{-Tupel über } \beta \wedge (\forall z \in \text{Z-POSS}(z_\omega))(\exists t' \in q(z))(t' \in \text{A-POSS}(t))\}.$$

Zu einem gesicherten Antworttupel t gibt es also in jeder möglichen Antwort ein Tupel t', so daß t als unvollständige Information von t' angesehen werden kann. Die Definition kann auch äquivalent wie folgt geschrieben werden:

$$\mathcal{GT}(q,z_\omega) =_{df} \{t \mid t\ \omega\text{-Tupel über } \beta \wedge (\forall z \in \text{Z-POSS}(z_\omega))(\exists t' \in \text{A-POSS}(t))(t' \in q(z))\}.$$

Beispiel 3.8: Seien in einem ω-Zustand z_ω die beiden Relationen R = {(1,1), (1,2), (1,3)} und S = {(1, ω)} aus Beispiel 3.5 gegeben, und sei $POSS_c$ als Z-POSS-Funktion gewählt. Für die Anfrage "R \ S" in z_ω erhalten wir {(1, ω), (ω,ω)} als Menge gesicherter Antworttupel. Für "S \ R" erhalten wir die leere Menge.
Für die beiden Ausdrücke "R" und "R ∪ S" betrachtet als Anfragen gilt $\mathcal{GT}$("R",z_ω) = $\mathcal{GT}$("R ∪ S",z_ω). Durch das Hinzufügen des Tupels (1, ω) zu R kommt keine gesicherte Information hinzu, da (1, ω) von den Tupeln aus R überdeckt wird und damit R eine mögliche Antwort für "R ∪ S" darstellt.

Da es sich bei der Überdeckungsrelation für ω-Tupel um eine partielle Ordnung handelt, folgt aus der Definition gesicherter Antworttupel unmittelbar, daß mit jedem Tupel t auch alle Tupel in einer Menge gesicherter Antworttupel enthalten sind, die von t bezüglich der zugehörigen Überdeckungsrelation überdeckt werden. Jede Menge gesicherter Antworttupel stellt somit einen endlichen Infimum-Halbverband dar mit der gegebenen Überdeckungsrelation als partieller Ordnung, der sich durch seine maximalen Elemente eindeutig charakterisieren läßt. Diese Eigenschaft nutzen wir zu einer ersten Definition gesicherter Antworten, in denen unbekannte Attributwerte erlaubt sind.

Definition: Sei q eine Anfrage vom Typ β an ein DB-Schema σ. Die gesicherte Maximum-Antwort auf q in einem ω-Zustand z_ω zu σ (in Zeichen: $\mathcal{GA}_{max}(q,z_\omega)$) ist die Menge der maximalen Elemente von $\mathcal{GT}(q,z_\omega)$.

Beispiel 3.9: Für "R \ S" mit R und S aus den Beispielen 3.5 und 3.8 erhalten wir {(1,ω)} als gesicherte Maximum-Antwort.

Die gesicherte totale Antwort auf eine Anfrage ist offensichtlich stets eine Teilmenge der gesicherten Maximum-Antwort. Beide Antworten stimmen überein, wenn die Menge aller gesicherten Antworttupel keine maximalen Tupel mit unbe-

kannten Attributwerten enthält. Letzteres gilt natürlich insbesondere dann, wenn die Anfrage an einen totalen Zustand gestellt wird.

Die gesicherte Maximum-Antwort auf die Anfrage "FIRMA" für den ω-Zustand aus Beispiel 3.3 (S. 39) mit $POSS_{ce}$ oder $POSS_{c}$ als einheitliche Möglichkeitsfunktion für den Zustand und die Antwort ist die Relation FIRMA selbst. Die Beschränkung auf maximale Tupel kann für andere Zustände aber zu unerwünschten Ergebnissen führen. Gibt es etwa in einer Relation R zwei Tupel t' und t, wobei t' eine Überdeckung von t ist, dann ist t nicht in der gesicherten Maximum-Antwort auf "R" enthalten, d.h. die Anfrage liefert nicht den Inhalt der Relation. Dieser Fall kann natürlich auch eintreten, wenn in Algebraanfragen Überdeckungen durch Anwendung von Projektionen erhalten werden.

Beispiel 3.10: Betrachte wieder die Relation FIRMA aus Beispiel 3.3. Für die Algebraanfrage "FIRMA [STADT, STRASSE]" erhalten wir {(Freudenstadt, Hauptstrasse)} als gesicherte Maximum-Antwort. Die Tupel, die sich aus den Projektionen für das erste und zweite Tupel der Relation ergeben, tragen nichts zum Ergebnis bei, da sie von der Projektion des letzten Tupels überdeckt werden.
Für diese Anfrage wäre die Aufnahme der beiden überdeckten Tupel in die gesicherte Antwort sicher naheliegender. Andererseits gibt es einen möglichen Zustand, in dem die Projektion nur das Tupel aus der gesicherten Maximum-Antwort liefert.

Wie wir gesehen haben, kann in gesicherten Maximum-Antworten durch das Ausblenden überdeckter Tupel ein Informationsverlust im Hinblick auf die Beschreibung möglicher Antworten auftreten. In der folgenden Definition wird von Antworten verlangt, daß sie ein gewisses Optimum bezüglich der Beschreibung möglicher Antworten darstellen. Als Sonderfall muß dabei allerdings der Fall ausgeklammert werden, daß eine der möglichen Antworten die leere Relation über dem Antworttyp darstellt und andere mögliche Antworten nicht leer sind. Die gewählte Bezeichnung *min/max-Antwort* erklären wir im Anschluß an die Definition.

Definition: Sei q eine Anfrage vom Typ β an ein DB-Schema σ. Eine gesicherte min/max-Antwort auf q in einem ω-Zustand z_ω zu σ (in Zeichen: $\mathcal{GA}_{min/max}(q,z_\omega)$) ist eine Teilmenge T von $\mathcal{GT}(q,z_\omega)$ mit folgenden Eigenschaften:
Im Fall $\mathcal{GT}(q,z_\omega) = \Phi$ ist T die leere Relation über β;
sonst gilt für T:
1) $(\forall z \in Z\text{-}POSS(z_\omega))(\exists u \in A\text{-}POSS(T))(u = q(z))$
2) $\neg(\exists T' \subseteq \mathcal{GT}(q,z_\omega))(T'$ hat die Eigenschaft 1 $\wedge$ $A\text{-}POSS(T') \subsetneq A\text{-}POSS(T))$
3) Keine echte Teilmenge von T erfüllt 1 und 2.

Beispiel 3.11: Betrachte einen Zustand z_ω mit einer Relation R über {A,B,C}.

R

A	B	C
3	4	5
3	ω	ω
ω	4	ω
ω	ω	5

Es gilt $\mathcal{GT}("R",z_\omega) = \{(x,y,z) \mid (3,4,5) \geq (x,y,z)\}$. Sei für A-POSS die enge Ausdehnung gewählt. Die Menge {(ω,ω,ω), (3,4,5)} erfüllt die erste Bedingung, nicht aber die zweite. Die Menge {(3,4,5), (3,ω,ω), (ω,4,5), (ω,4,ω), (ω,ω,5), (3,ω,5)} erfüllt die

ersten beiden Bedingungen, aber nicht die dritte Bedingung, da die Elemente (ω,4,5) und (3,ω,5) aus ihr entfernt werden können, ohne daß die ersten beiden Eigenschaften verlorengehen. Es gibt nur eine gesicherte min/max-Antwort: R selbst. Betrachte nun die Relation R' = R ∪ {(ω,4,5)} anstelle von R. Die eindeutige gesicherte min/max-Antwort ist in diesem Fall wiederum R, d.h. wir "verlieren" das hinzugekommene Tupel und erhalten damit nicht die Identität.

Allgemein gilt für gesicherte min/max-Antworten mit der engen Ausdehnung als A-POSS-Funktion, daß sie kein Tupel t enthalten können, so daß es zwei weitere von t verschiedene Tupel t' und t'' gibt mit t' ≥ t ≥ t''. Anders formuliert besteht eine solche Antwort aus Tupeln, die bezüglich der Überdeckungsrelation in der Antwort minimal oder maximal sind (daher die Bezeichnung min/max-Antwort).

Bemerkung: Für den Fall, daß keine mögliche Antwort leer ist, gewährleistet die erste Bedingung in der Definition, daß jede mögliche Antwort als mögliche Relation der gesicherten Antwort erhalten werden kann. Diese Bedingung entspricht der Eigenschaft *adequate* (hinreichend), die in [Bis83] von Operationen auf ω-Relationen gefordert wird. Sie ist im allgemeinen nicht mit der Vervollständigung sondern nur mit der Ausdehnung oder der engen Ausdehnung als POSS-Funktion zu erfüllen. Die zweite Bedingung verlangt, daß die Menge der möglichen Relationen zur Antwort eine minimale darstellbare Obermenge der Menge der möglichen Antworten ist. Diese Forderung entspricht der Bedingung *restricted* (beschränkt), die in [Bis83] an Ergebnisse von Operationen gestellt wird. Da jede enge Ausdehnung auch eine Ausdehnung ist, eignet sich somit nur die enge Ausdehnung als POSS-Funktion für Antworten mit dieser Definition.

Das nächste Beispiel zeigt uns, daß gesicherte min/max-Antworten nicht eindeutig bestimmt sind.

Beispiel 3.12: Sei z_ω ein Zustand mit folgender Relation R vom gleichen Typ wie in Beispiel 3.11:

R

A	B	C
1	ω	5
3	4	5
ω	ω	6

Betrachte die Anfrage

q = "R[(A = 1 ∧ B = 4) ∨ C = 5]".

$\mathcal{QT}(q,z_\omega)$ besteht aus allen Tupeln, die von (1,ω,5) oder (3,4,5) überdeckt werden. In einer gesicherten min/max-Antwort auf q muß es ein Tupel geben, das von (1,4,6) überdeckt wird, da einige mögliche Antworten das Tupel (1,4,6) enthalten. Es sind zwei Tupel in $\mathcal{QT}(q,z_\omega)$ enthalten, die von (1,4,6) überdeckt werden und bezüglich dieser Eigenschaft maximal sind: (1,ω,ω) und (ω,4,ω). Damit erhalten wir zwei gesicherte min/max-Antworten, die überdeckungsäquivalent sind:

{(1,ω,5), (3,4,5), (1,ω,ω)} und {(1,ω,5), (3,4,5), (ω,4,ω)}.

Die Mengen der möglichen Relationen zu beiden Antworten sind offensichtlich für alle betrachteten POSS-Funktionen verschieden.

Die Menge der Teilmengen von $\mathcal{GT}(q,z_\omega)$, die die Eigenschaft 1 erfüllen, kann partiell geordnet werden: $T \leq T'$ falls A-POSS(T) $\subseteq$ A-POSS(T').
Diejenigen Mengen, die die Bedingung 2 erfüllen, sind die Minima bezüglich dieser Ordnung. Zwei Minima können die gleiche Menge möglicher Relationen haben. Für jede Menge solcher Minima mit gleicher Menge möglicher Relationen gibt es ein eindeutig bestimmtes kleinstes Element bezüglich der Mengeninklusion ([Bis83]). Die dritte Bedingung wählt dieses Element aus.

Wir haben in den Beispielen gesehen, daß sich die Bedeutung von Vorkommen des Symbols ω in gesicherten Antworten von der Bedeutung unterscheiden kann, die den Vorkommen von ω in Zuständen unterliegt: Ein Vorkommen von ω in einem Tupel einer gesicherten Antwort bedeutet nicht notwendigerweise, daß an dieser Stelle ein beliebiger Wert eingesetzt werden kann. In Beispiel 3.9 (S. 42) etwa können nur die Werte 1, 2 und 3 für das Vorkommen von ω in dem Antworttupel $(1,\omega)$ eingesetzt werden. Die folgende Form einer gesicherten Antwort verlangt, daß Vorkommen von ω in Zuständen und Antworten insofern die gleiche Interpretation unterliegt, als keine Einschränkung in Antworten bezüglich einsetzbarer Werte besteht (daher auch die Bezeichnung *uni* für universell). Diese Antwortform geht von der Unabhängigkeit aller Vorkommen unbekannter Werte in Antworten aus.

Definition: Sei q eine Anfrage vom Typ β an eine DB-Schema σ. Die strenge gesicherte uni-Antwort auf q in einem ω-Zustand z_ω zu σ (in Zeichen: $\mathcal{GA}_{s\text{-}uni}(q,z_\omega)$) ist die Menge der ω-Tupel t von $\mathcal{GT}(q,z_\omega)$ mit folgender Eigenschaft:

$(\forall t')(t' \trianglerighteq t \Rightarrow (\exists z \in Z\text{-}POSS(z_\omega))(t' \in q(z)))$.

Beispiel 3.13: Für "R \ S" mit R und S aus den Beispielen 3.5 (S. 40) und 3.8 (S. 42) erhalten wir eine leere strenge gesicherte uni-Antwort.
Für die Anfrage nach dem Inhalt von R aus Beispiel 3.11 (S. 43) ergibt sich die strenge gesicherte uni-Antwort als die Menge der gesicherten Antworttupel ohne das Tupel (ω,ω,ω).

Die Definition strenger gesicherter uni-Antworten enthält keine Bedingung bezüglich der Minimalität der Antwort. Daher hat jede solche Antwort die folgende Eigenschaft: Falls zwei Tupel t' und t mit $t' \geq t$ in einer strengen gesicherten uni-Antwort enthalten sind, dann sind auch alle Tupel t" mit $t' \geq t'' \geq t$ in der Antwort enthalten. Mit der engen Ausdehnung als POSS-Funktion für Antworten kann die Antwort daher wegen Lemma 2.3 ohne Informationsverlust auch auf die bezüglich der Überdeckungsrelation minimalen und maximalen Elemente beschränkt werden.

Die strenge gesicherte uni-Antwort betrachtet verschiedene Vorkommen von ω in einem Antworttupel als unabhängig voneinander. Durch eine Anfrageformulierung können aber Abhängigkeiten von Werten in Antworttupeln bedingt sein, so daß diese Antwortform häufig zu einschränkend ist. Außerdem ist die Absicherung der Unabhängigkeit im allgemeinen mit einem Aufwand verbunden, der keine effiziente Anfrageauswertung erlaubt.

Ein Trivialbeispiel mag zeigen, daß die strenge gesicherte uni-Antwort dann zu restriktiv ist, wenn Abhängigkeiten zwischen unterschiedlichen Vorkommen von ω in der Antwort bestehen.

Beispiel 3.14: Sei R eine Relation, in der nur das Tupel $(1,\omega)$ vorkommt. Betrachte folgende TRC-Anfrage: $(x,y)/\ R\ x \wedge R\ y$. Die strenge gesicherte uni-Antwort auf

diese Anfrage ist leer, da nicht für alle Überdeckungen von (1,ω,1,ω) eine mögliche Antwort existiert, in der diese Überdeckung enthalten ist. Wähle etwa das Tupel (1,2,1,3).

Um solche Abhängigkeiten zu berücksichtigen, verlangen wir in der folgenden Definition nur, daß zu jedem Vorkommen von ω in einem Antworttupel zu jedem Wert, der für ω eingesetzt werden kann, eine Überdeckung existiert, die in einer möglichen Antwort enthalten ist.

Definition: Sei q eine Anfrage vom Typ β an ein DB-Schema σ. Die gesicherte uni-Antwort auf q in einem ω-Zustand z_ω zu σ (in Zeichen: $\mathcal{GA}_{uni}(q,z_\omega)$) ist die Menge der ω-Tupel t von $\mathcal{GT}(q,z_\omega)$ mit folgender Eigenschaft:

$$(\forall A \in \beta)(\forall t' \in \Re_\beta)(t'_{|\{A\}} \geq t_{|\{A\}} \Rightarrow (\exists t'' \in \Re_\beta)(t''(A) = t'(A) \wedge t'' \trianglerighteq t \wedge (\exists z \in Z\text{-}POSS(z_\omega))(t'' \in q(z))))$$

Beispiel 3.15: Die gesicherte uni-Antwort auf die Anfrage in Beispiel 3.14 ist {(1,ω,1,ω)}. Für die Differenz R\S mit den Relationen R = {(1,1), (1,2), (1,3)} und S = {(1,ω)} aus Beispiel 3.8 (S. 42) erhalten wir eine leere Antwort als gesicherte uni-Antwort, da alle Überdeckungen des gesicherten Antworttupels (1,ω), die nicht in R vorkommen, in keiner möglichen Antwort enthalten sind.
Fragen wir nach dem Inhalt der Relation R aus Beispiel 3.11 (S. 43), dann sind bis auf (ω,ω,ω) alle gesicherten Antworttupel zu "R" in der gesicherten uni-Antwort enthalten.

Die Beispiele zeigen einige Nachteile der gesicherten uni-Antwort:

- Falls Vorkommen von ω in gesicherten Antworttupeln nicht Vorkommen von ω im Zustand entsprechen, sondern wie bei der Differenz in Beispiel 3.15 "Platzhalter" für eine endliche Anzahl von definierten Werten aus dem Zustand darstellen, geht Information verloren.
- Falls es Tupel t' und t in der gesicherten uni-Antwort gibt mit t' ≥ t, dann sind auch alle Tupel t" mit t' ≥ t" ≥ t in der gesicherten uni-Antwort enthalten (vgl. Beispiel 3.11, S. 43).

Der zweite genannte Nachteil bedeutet, daß auch bei dieser Antwortform kein Informationsverlust auftritt, wenn nur die bezüglich der Überdeckungsrelation minimalen und maximalen Elemente in der Antwort aufgeführt werden.

Trotz der genannten Nachteile ist die gesicherte uni-Antwort wichtig im Hinblick auf Auswertungsverfahren, da einfache Verfahren Antworten liefern, die stets in der entsprechenden gesicherten uni-Antwort enthalten sind, aber nicht immer in der gesicherten max- oder min/max-Antwort.

Die bisher betrachteten Antwortformen stellen DB-Relationen in erster Normalform dar, in denen als Attributwerte nur gewöhnliche Werte aus den Wertebereichen und das spezielle Symbol ω verwendet werden und in denen nur Tupel aus der Menge gesicherter Antworttupel vorkommen dürfen. Durch diese Beschränkungen kann auf verschiedene Weise gesicherte Information verlorengehen.

Zunächst enthalten die Tupel in der Menge gesicherter Antworttupel keine Information darüber, welche Werte die Vorkommen von ω in einem Tupel in den zugehörigen Überdeckungen der möglichen Antworten annehmen können. In Beispiel 3.5 (S. 40) geht etwa die Information verloren, daß ω in (1,ω) nur die Werte 1, 2 und 3 annehmen kann, wobei in einer möglichen Antwort auch mehrere Überdeckungen von (1,ω) vorkommen können.

Ein weiterer Informationsverlust kann bei gesicherten uni-Antworten eintreten, da in Antworten dieser Form nur Tupel aus der Menge gesicherter Antworttupel übernommen werden, in denen ein Vorkommen von ω wie die Vorkommen von ω in Zuständen durch beliebige Werte des Wertebereichs ersetzt werden kann.

Eine naheliegende Idee zur Verbesserung des Informationsgehalts gesicherter Antworten ist, Tupel mit mengenwertigen Attributen statt nur flache Tupel zuzulassen und die Werte in diesen Attributen als "oder-Mengen" ([IV89], [INV91]) zu betrachten. Damit bestimmt jedes Tupel mit derartigen Attributwerten eine Menge flacher Tupel. Die Realisierbarkeit solcher Darstellungsformen hängt natürlich von der Mächtigkeit der beteiligten Wertemengen ab.

3.2.2 Formen gesicherter Antworten für Anfragen an V-Zustände

Für V-Zustände ist eine Belegungsfunktion die naheliegende Form der Möglichkeitsfunktion. Für Antworten bietet sich an, die Variablen aus den Zuständen wann immer möglich wie Konstanten als Attributwerte zu verwenden und damit ebenfalls eine Belegungsfunktion als Möglichkeitsfunktion herzunehmen. Die Variablen in einer Antwort geben eindeutig Stellen in den Relationen des betrachteten Zustands an, und erlauben es somit, eineindeutige Verknüpfungen zwischen Vorkommen unbekannter Werte in Antworten und Zuständen herzustellen. Wir haben aber auch gesehen (Beispiel 3.7, S.41), daß es für die Darstellung gesicherter Information in Antworten erforderlich sein kann, neue Variable zu verwenden, das heißt Variable, die im betrachteten Zustand nicht vorkommen. Statt der Wahl neuer Variablen bietet sich auch der Übergang zu einer Mischform von V- und ω-Relationen an.

Variable, die in einer Antwort und dem betrachteten V-Zustand vorkommen, müssen den gleichen unbekannten Wert bezeichnen, um Interpretationsprobleme zu vermeiden. Daher muß in Definitionen von Antworten die Belegungsfunktion für eine Antwort auf allen Variablen, die im Zustand vorkommen, mit der Belegungsfunktion für den Zustand übereinstimmen. Dies führt unmittelbar zu einer ersten Antwortform, die mit der gesicherten uni-Antwort bei ω-Zuständen vergleichbar ist. Sie wurde schon in [Lip84] erwähnt und dort als "CWA response" bezeichnet.

Definition: Sei q eine Anfrage vom Typ β an ein DB-Schema σ. Die gesicherte uni-V-Antwort auf q in einem V-Zustand z_V zu σ (in Zeichen: $\mathcal{GA}_{uni}(q,z_V)$) ist wie folgt definiert:

$$\mathcal{GA}_{uni}(q,z_V) =_{df} \{t \mid t \text{ V-Tupel über } \beta \wedge (\forall \nu: \nu \text{ Belegung für } \beta,V)(\nu(t) \in q(\nu(z_V)))\}.$$

Ist t total, dann muß t in jeder möglichen Antwort auf q im Zustand z_V enthalten sein; die gesicherte totale Antwort ist demnach stets in der gesicherten uni-V-Antwort enthalten.

Aus der Definition ergibt sich sofort, daß die Anfrage nach dem Inhalt einer Relation immer die Relation selbst liefert.

Beispiel 3.16: Sei in einem V-Zustand die folgende V-Relation R enthalten:

R

A	B	C
2	ω_1	4
3	ω_2	3
2	1	5

Für die Anfrage "R[A ≠ 3]" erhalten wir $\{(2,\omega_1,4), (2,1,5)\}$ als gesicherte uni-V-Antwort.
Für den Ausdruck "(R \ $\{(2,\omega_3,\omega_4)\}$)[A,B]" betrachtet als Anfrage ist $\{(3,\omega_2)\}$ die gesicherte uni-V-Antwort. Die Information, daß in jeder möglichen Antwort ein Tupel $(2,\nu(\omega_1))$ oder (2,1) oder beide Tupel enthalten sind, kann mit dieser Antwortform nicht repräsentiert werden.

In einer gesicherten uni-V-Antwort kann keine Variable vorkommen, die nicht auch im betrachteten V-Zustand vorkommt. Für eine derartige Variable könnte eine Belegungsfunktion angegeben werden, die für alle Variablen des Zustands einen anderen Wert als für die Variable liefert, wobei dieser Wert auch noch von allen Konstanten im Zustand verschieden ist. Es werden daher andere Attributwerte benötigt. Zwei Vorgehensweisen bieten sich an:

- Es werden "neue" Variable eingeführt, die nicht in V oder zumindest nicht in dem betrachteten V-Zustand vorkommen.
- Es wird wieder ein spezielles Symbol, etwa ω, verwendet.

Betrachten wir im folgenden zunächst die erste Möglichkeit. Als erstes ist die Bedeutung der neuen Variablen festzulegen. Während Variable aus V stets genau einen Wert repräsentieren, muß die Bedeutung der neuen Variablen im Zusammenhang mit der Darstellung möglicher Antworten durch gesicherte Antworten gesehen werden. Wir haben es mit einer ähnlichen Fragestellung zu tun, wie sie sich mit der Vervollständigung als POSS-Funktion für ω-Zustände stellt. Betrachten wir dazu ein Beispiel (vergleiche mit Beispiel 3.5, S. 40).

Beispiel 3.17: Seien die beiden folgenden V-Relationen R und S eines V-Zustands gegeben:

R

A	B	C
1	1	1
1	2	2
1	3	3

S

A	B	C
1	ω_1	ω_2

Betrachte den Ausdruck "R \ S". Gesicherte Information ist, daß mindestens zwei Tupel in jeder möglichen Antwort auf die Anfrage "R \ S" enthalten sind. Da die Werte dieser Tupel in B und C gleich sein müssen, bieten sich folgende Mengen für eine gesicherte Antwort an: $\{(1,\psi_1,\psi_1), (1,\psi_2,\psi_2)\}$ oder $\{(1,\psi_1,\psi_1)\}$; dabei sind ψ_1 und ψ_2 Variable, die nicht in V oder zumindest nicht im betrachteten V-Zustand vorkommen.
Will man zur möglichen Antwort R gelangen, muß in der ersten Antwort eine der beiden Variablen durch zwei Werte ersetzt werden; in der zweiten Antwort muß eine Ersetzung durch drei Werte erfolgen. Die neuen Variablen sollten daher nicht als Repräsentanten genau eines Wertes gelesen werden, wenn an die Beschreibung möglicher Antworten durch gesicherte Antworten gedacht wird. Ohne zusätzliche Angaben tragen die beiden Tupel der ersten Antwort offensichtlich die gleiche Information unabhängig davon, wie die neuen Variablen interpretiert werden.

Wir erweitern zunächst die genaue Überdeckungsrelation für V-Tupel auf Tupel mit Variablen aus zwei disjunkten Mengen, wobei Elemente einer der beiden

Mengen (V) als informativer angesehen werden als die Elemente der anderen Menge (V'). Die Verträglichkeit mit der Definition für die genaue Überdeckung ist aus der folgenden Definition direkt abzulesen.

Definition: Seien V und V' abzählbar unendliche Mengen von Variablen mit $V \cap V' = \emptyset$. Sei ein $V \cup V'$-Tupel über $\alpha_i \subseteq \alpha$ analog zu einem V-Tupel definiert mit $V \cup V'$ als Variablenmenge statt V.
Ein $V \cup V'$-Tupel t' repräsentiert mindestens die Information, die das $V \cup V'$-Tupel t repräsentiert (t' ist informativer als t, in Zeichen: $t' \geq_i t$) $\Leftrightarrow_{df}$

1) $(\forall \nu$: ν Belegung für $\alpha, V \cup V')(\exists \nu'$: ν' Belegung für $\alpha, V \cup V')(\nu(t') = \nu'(t))$

2) $t(A) \in V \cup DOM_\alpha \Rightarrow (t(A) = t'(A) \vee t'(A) \in DOM_\alpha)$ für alle $A \in \alpha_i$

Repräsentiert ein $V \cup V'$-Tupel t' mindestens die Information, die t repräsentiert, und umgekehrt, dann heißen t' und t informationsäquivalent (in Zeichen: $t' \equiv_i t$).

Seien die Variablen aus V' zu Unterscheidung von den Variablen aus V als indizierte Formen von ψ geschrieben.

Beispiele: $(1,\omega_1,4) \geq_i (1,\omega_1,\omega_3)$, $(1,\omega_1,4) \geq_i (1,\psi_1,\omega_3)$, $(1,\omega_1,\omega_1) \geq_i (\psi_1,\omega_1,\omega_1)$, $(\psi_2,\omega_1,\omega_1) \geq_i (\psi_2,\psi_3,\omega_1)$, $(\psi_1,\psi_1) \geq_i (\psi_1,\psi_2)$

und

$(1,\omega_1,\psi_1) \equiv_i (1,\omega_1,\psi_2)$, $(\psi_1,\psi_2) \equiv_i (\psi_1,\psi_3)$,

Werden in zwei informationsäquivalenten Tupeln alle Variablen aus V' durch das gleiche Symbol, etwa ω, ersetzt, dann sind die resultierenden Tupel identisch.

Mit dieser Ordnungsrelation können wir nun das Gegenstück zum Verschmelzen von V-Tupeln (*merge*-Operation, s. Kapitel 2, S. 28) definieren, die *meet*-Operation. Im Ergebnis der meet-Operation soll die Information enthalten sein, die gegebenen Tupeln gemeinsam ist. Dabei können wir uns bei der Definition nicht allein auf die Ordnungsrelation stützen und in Analogie zu einer Infimumbildung verfahren, da die Variablen aus V nicht wie die Variablen aus V' frei verwendet werden dürfen. Diese Beschränkung ergibt sich aus der "Anbindung" von Variablen aus V an Stellen im Zustand. Folgendes Beispiel zeigt, worauf zu achten ist:
Betrachte die beiden V-Tupel $(1,2,\omega_1)$ und $(\omega_2,2,\omega_3)$. Ein bezüglich $\geq_i$ maximales $V \cup V'$-Tupel, das von beiden Tupeln überdeckt wird, ist $(\omega_2,2,\psi_1)$. Dieses Tupel repräsentiert aber in der ersten Komponenten eine Information, die 1 und ω_2 nur dann gemeinsam ist, wenn beide Werte identisch sind.

Da wir die Operation später auf Mengen von V-Tupeln anwenden wollen, definieren wir sie gleich entsprechend.

Definition: Sei T eine endliche Menge von V-Tupeln über der gleichen Attributmenge α. Dann ist meet(T) die Menge aller $V \cup V'$-Tupel t'' über α mit folgenden Eigenschaften:

1) $t''(A) = \begin{cases} t(A) & \text{für ein } t \in T, \text{ falls } (\forall t,t' \in T)(t(A) = t'(A)) \\ \psi_i & \psi_i \in V', \quad \text{sonst} \end{cases}$

2) t'' ist das bezüglich $\geq_i$ maximale Tupel mit Eigenschaft 1, für das gilt: $(\forall t \in T)(t \geq_i t'')$

Beispiele: $meet(\{(1,2,\omega_1), (\omega_2,2,\omega_3)\}) = \{(\psi_i,2,\psi_j) \mid \psi_i,\psi_j \in V', i \neq j\}$

$meet(\{\omega_1,2,\omega_1), (\omega_2,3,\omega_2)\}) = \{(\psi_i,\psi_j,\psi_i) \mid \psi_i,\psi_j \in V', i \neq j\}$

Die meet-Operation liefert zwar im allgemeinen eine unendliche Menge von V∪V'-Tupeln, diese Tupel sind aber alle informationsäquivalent.

Lemma 3.1: Sei M eine endliche Menge von V-Tupeln. Dann sind alle Elemente aus meet(M) informationsäquivalent.

Beweis: Wegen der Eigenschaft 1 können sich Tupel aus T nur in Attributen unterscheiden, in denen sie Werte aus V' haben. Zwei Tupel über einer gemeinsamen Attributmenge, die als Attributwerte nur Variable aus V' haben, sind genau dann informationsäquivalent, wenn es eine Permutation von den in ihnen vorkommenden Variablen gibt, die sie ineinander überführt. Dies folgt unmittelbar aus der Definition von $\geq_i$, da bei Belegungen alle Variablen gleichrangig sind.
Tritt in einem $t'' \in meet(T)$ eine Variable in zwei Attributen A und B auf, dann müssen in allen Tupeln von T die Werte in A und B ebenfalls übereinstimmen, da sonst die Bedingung $(\forall t \in T)(t \geq_i t'')$ in der Definition von meet nicht erfüllt sein kann: Es gibt dann ein mögliches Tupel zu einem Tupel t aus T, das kein mögliches Tupel zu t'' ist. Umgekehrt muß jede Gleichheit aller Tupel von T in zwei Attributen in einem Tupel t'' von meet(T) durch Gleichheit der Werte in diesen Attributen repräsentiert werden, wenn t'' maximal bezüglich $\geq_i$ sein soll. Daraus folgt, daß die erwähnte Permutation für jedes Paar von Tupeln aus meet(T) existiert. Damit sind alle Tupel aus meet(T) informationsäquivalent. □

Für die Definition gesicherter Antworttupel müssen wir verlangen, daß in Antworttupeln für alle vorkommenden Variablen aus V gilt, daß sie im entsprechenden Zustand vorkommen und daß sie den gleichen unbekannten Wert wie die Vorkommen der Variablen im Zustand repräsentieren.

Zur Vereinfachung der Sprechweise wollen wir die Wirkung von Belegungsfunktionen auf V∪V'-Tupel mit entsprechender Notation variieren können:
Mit einer Belegung für α,V (Belegung für α,V') werden bei Anwendung auf ein V∪V'-Tupel alle Variablen aus V' (V) wie Konstanten aus DOM_α betrachtet.
Damit können die Ersetzungen von Variablen aus V und V' getrennt vorgenommen werden.

Definition: Seien q eine Anfrage vom Typ β an ein DB-Schema σ und z_V ein V-Zustand zu σ. Sei V' wie bisher.
Die Menge aller gesicherten Antworttupel zu q im Zustand z_V (in Zeichen: $\mathcal{GT}(q,z_V)$) ist definiert als

$$\mathcal{GT}(q,z_V) =_{df} \{t \mid t \text{ V}\cup\text{V'-Tupel über } \beta \wedge (\forall \nu\text{: } \nu \text{ Belegung für } \alpha\cup\beta,V)(\exists \nu'\text{: } \nu' \text{ Belegung für } \beta,V')(\nu'(\nu(t)) \in q(\nu(z_V)))\}.$$

Offensichtlich ist die gesicherte uni-V-Antwort stets in der entsprechenden Menge aller gesicherten Antworttupel enthalten: Betrachte nur die Tupel mit Variablen aus V.

Falls $q(\nu(z_V))$ für jede Belegung ν nicht leer ist, enthält die Menge aller gesicherten Antworttupel unendlich viele Tupel. Dies folgt aus der Unendlichkeit von V'. Nur eine endliche Anzahl dieser Tupel ist aber nicht informationsäquivalent.

Lemma 3.2: Für alle Anfragen q an ein DB-Schema σ und alle V-Zustände z_V zu σ enthält $\mathcal{GT}(q,z_V)$ höchstens endlich viele Tupel, die nicht informationsäquivalent sind.

Beweis: Die Anzahl der Variablen und Konstanten in z_V ist endlich. Da alle Variablen aus V und alle Konstanten, die in Tupeln von $\mathcal{GT}(q,z_V)$ auftreten, auch im

Zustand oder der Anfrage vorkommen müssen, gibt es nur endlich viele Tupel in $\mathcal{GT}(q,z_V)$, die sich in den Attributen, in denen ihre Werte Konstanten oder Variable aus V sind, voneinander unterscheiden.
Betrachte eine beliebige Menge M von V∪V'-Tupeln über einer gemeinsamen Attributmenge. Kommen in den Tupeln von M nur Variable aus V' vor, dann gibt es nur endlich viele Tupel in M, die unter allen Permutationen von V' verschieden voneinander sind. Wegen $(\psi_h) \equiv_i (\psi_j)$ für beliebige Variablen ψ_h, ψ_j aus V' gibt es damit auch nur endliche Teilmengen von M mit nicht informationsäquivalenten Tupeln. □

Betrachten wir nun die Quotientenmenge $\mathcal{GT}(q,z_V)/_{\equiv_i}$ zusammen mit der partiellen Ordnung $[t'] \geq_i [t] \Leftrightarrow_{df} t' \geq_i t$.
Diese Struktur ist ein Infimum-Halbverband mit $[(\psi_1,\psi_2,....,\psi_{|\beta|})]$, $\psi_i \neq \psi_j$ für $i,j \in \{1,....,|\beta|\}$, $i \neq j$, als minimalem Element, falls $\mathcal{GT}(q,z_V)$ nicht leer ist. Ersetzen wir alle Vorkommen von Variablen aus V' in den Tupeln durch ω, dann enthält jede Klasse nur noch ein Element und einige Klassen werden identisch.

Beispiel 3.18: Für die Anfrage aus Beispiel 3.7 (S. 41) erhalten wir als Quotientenmenge $\mathcal{GT}(q,z_V)/_{\equiv_i}$ $\{[(2,\psi_1)], [(\psi_2,\psi_1)]\}$.

Wir können nun ähnliche Formen gesicherter Antworten definieren, wie wir es für Anfragen in ω-Zuständen getan haben. Die Variablen eröffnen uns dabei weitergehende Möglichkeiten: Durch Aufnahme informationsäquivalenter Tupel können mit gesicherten Antworten mögliche Antworten genauer beschrieben werden. Bei der Auswahl von Tupeln der Klassen einer Quotientenmenge $\mathcal{GT}(q,z_V)/_{\equiv_i}$ sollte dabei so vorgegangen werden, daß kein Verlust bezüglich der durch die Variablen aus V' repräsentierten Information erfolgt. Dies bedeutet, daß die Anzahl verschiedener Variablen aus V', die insgesamt in den gewählten Repräsentanten vorkommen, möglichst gering sein sollte.

Wie bei der Menge gesicherter Antworttupel zu Anfragen in ω-Zuständen ist jede Quotientenmenge $\mathcal{GT}(q,z_V)/_{\equiv_i}$ durch Angabe der maximalen Elemente eindeutig bestimmt. Anstatt jedes maximale Element durch genau einen Repräsentanten in einer gesicherten Maximum-Antwort zu repräsentieren, kann die Auswahl mehrerer Elemente aus einer Klasse dazu genutzt werden, durch gesicherte Antworten mögliche Antworten besser zu beschreiben. Im folgenden definieren wir eine einfache Form einer gesicherten Maximum-Antwort.

Definition: Sei q eine Anfrage vom Typ β an ein DB-Schema σ. Eine gesicherte Maximum-V-Antwort auf q in einem V-Zustand z_V zu σ (in Zeichen: $\mathcal{GA}_{max}(q,z_V)$) ist eine Menge T von Repräsentanten aller maximalen Elemente von $\mathcal{GT}(q,z_V)/_{\equiv_i}$, so daß gilt:

$$(\forall \nu\text{: } \nu \text{ Belegung für } \alpha\cup\beta,V)(\exists \nu'\text{: } \nu' \text{ Belegung für } \beta,V')(\forall t \in T)(\nu'(\nu(t)) \in q(\nu(z_V))).$$

Die Bedingung, die an die Auswahl der Repräsentanten gestellt wird, sichert ab, daß über Variable aus V' keine unzulässige Identifikation von Attributwerten in den Tupeln einer Antwort erfolgen kann. Eine solche Identifikation ist unzulässig im Sinne einer gesicherten Information, wenn es zu den betroffenen Tupeln eine Belegung ν gibt, so daß keine Belegung ν' gefunden werden kann, die die beiden Tupel in die durch ν gegebene mögliche Antwort abbildet.

Falls die Auswahl der Repräsentanten so erfolgt, daß insgesamt die Anzahl unterschiedlicher Variablen aus V' minimal ist, ist eine gesicherte Maximum-Antwort bis

auf systematische Umbenennung eventuell vorkommender Variablen aus V' eindeutig bestimmt.

Beispiel 3.19: Betrachte die Anfrage "R \ S" für die beiden folgenden Relationen (in diesem und folgenden Beispielen kommen V-Relationen vor, in denen Variable mehrfach auftreten; dadurch geraten wir nicht in Konflikt mit unserer Annahme, daß solche mehrfachen Vorkommen von Variablen in Zustandsrelationen ausgeschlossen sind, da die Beispielrelationen durch entsprechende Ergänzung des jeweils betrachteten Ausdrucks aus V-Zuständen erzeugt werden können):

R

A	B	C
1	2	4
1	2	5
1	3	4
1	3	5

S

A	B	C
1	ω_1	4
1	ω_1	5

Als maximale Elemente der Menge aller gesicherten Antworttupel erhalten wir $\{(1,\psi_i,4) \mid \psi_i \in V'\} \cup \{(1,\psi_i,5) \mid \psi_i \in V'\}$. In jeder möglichen Antwort sind mindestens zwei Tupel mit gleichem Wert im Attribut B enthalten. Damit kann in den Repräsentanten der beiden Klassen die gleiche Variable aus V' in B verwendet werden. Als gesicherte Maximum-Antwort mit minimaler Anzahl von Variablen aus V' können wir damit wählen: $\{(1,\psi_1,4), (1,\psi_1,5)\}$. Als in diesem Sinne nicht minimale Antwort ist aber auch $\{(1,\psi_1,4), (1,\psi_2,5)\}$ eine gesicherte Maximum-Antwort.

Beim Übertragen der gesicherten min/max-Antworten auf Anfragen in ω-Zuständen auf V-Zustände muß entweder die Belegungsfunktion so abgeändert werden, daß Variable aus V' durch Mengen von Werten belegt werden können oder die Forderung nach Darstellbarkeit aller möglichen Antworten muß in der Definition geeignet berücksichtigt werden, wie es in der folgenden Definition geschieht.

Definition: Sei q eine Anfrage vom Typ β an ein DB-Schema σ. Eine gesicherte min/max-V-Antwort auf q in einem V-Zustand z_V zu σ (in Zeichen: $\mathcal{GA}_{min/max}(q,z_V)$) ist eine Teilmenge T von Repräsentanten der Klassen von $\mathcal{GT}(q,z_V)/\equiv_i$ mit folgenden Eigenschaften:

Im Fall $\mathcal{GT}(q,z_V) = \Phi$ ist T die leere Relation über β;

sonst gilt:

1) $(\exists S \subseteq \{s \mid s \in [t] \wedge t \in T\})(\forall \nu$: ν Belegung für $\alpha \cup \beta, V)(\exists \nu'$: ν' Belegung für $\beta, V')$
$(\nu'(\nu(S)) = q(\nu(z_V)))$

2) $\neg$ (∃T': T' Teilmenge von Repräsentanten der Klassen von $\mathcal{GT}(q,z_V)/\equiv_i$)
(T' hat die Eigenschaft 1 $\wedge$ $\{\nu(T') \mid \nu$ Belegung für $\beta, V \cup V'\} \subsetneq$
$\{\nu(T) \mid \nu$ Belegung für $\beta, V \cup V'\}$)

3) keine echte Teilmenge von T erfüllt 1 und 2

Die Bemerkungen, die wir zu gesicherten min/max-Antworten für Anfragen an ω-Zustände gemacht haben, gelten sinngemäß auch hier. Bedingung 2 gewährleistet zusammen mit Bedingung 3, daß sowohl die Anzahl der Repräsentanten als auch die Anzahl der Variablen aus V' minimal ist.

Beispiel 3.20: Seien die beiden folgenden V-Relationen gegeben:

R

A	B	C
3	4	5
3	5	ω_1
3	6	5
2	5	5

S

A	B	C
3	ω_2	5
ω_3	ω_2	5

Betrachte wieder die Differenz "R \ S" als Anfrage. Als gesicherte min/max-V-Antwort erhalten wir $\{(3,\psi_1,5), (3,\psi_2,\psi_3), (\psi_4,\psi_2,5)\}$. In der gesicherten max-V-Antwort ist nur das Tupel $(3,\psi_1,5)$ enthalten.

Anstatt neue Variable in Antworten zu verwenden, kann auch eine einfachere Darstellung gewählt werden, indem nur ein einziges neues Symbol anstatt unterschiedlicher Variablen verwendet wird. Sei wieder ω als dieses Symbol gewählt. Antworten bestehen dann aus (ω,V)-Tupeln, das heißt aus Tupeln, in denen sowohl Variable aus dem betrachteten V-Zustand als auch der spezielle Wert ω vorkommen.

Für die Definition neuer Antwortformen, die sich mit dieser gemischten Tupelform angeben lassen, benötigen wir eine Erweiterung der Belegungsfunktionen: Der Argumentbereich der Belegungsfunktionen wird ausgedehnt auf $V \cup \{\omega\}$, wobei ν auf $\{\omega\}$ wie auf den Konstanten die Identität sein soll. Wie oben definieren wir zunächst gesicherte Antworttupel.

Definition: Sei q eine Anfrage vom Typ β an ein DB-Schema σ im V-Zustand z_V. Die Menge aller gesicherten (ω,V)-Antworttupel zu q im Zustand z_V (in Zeichen: $\mathcal{GT}_\omega(q,z_V)$) ist definiert als

$$\mathcal{GT}_\omega(q,z_V) =_{df} \{t \mid t \text{ } (\omega,V)\text{-Tupel über } \beta \wedge (\forall \nu\text{: } \nu \text{ Belegung für } \alpha, V\cup\{\omega\}) (\exists t' \in q(\nu(z_V)))(t' \trianglerighteq \nu(t))\}.$$

Da $\nu(t)$ ein ω-Tupel ist, steht $\trianglerighteq$ für die Vervollständigung von ω-Tupeln.

Beispiel 3.21: Für die Anfrage in Beispiel 3.7 (S.41) erhalten wir $(2,\omega)$ und (ω,ω) als gesicherte (ω,V)-Antworttupel.
Für die erste Anfrage in Beispiel 3.16 (S. 47) erhalten wir alle Tupel als gesicherte (ω,V)-Tupel, die von den beiden V-Tupeln der gesicherten uni-V-Antwort genau überdeckt werden (zu genauen Überdeckungen bei (ω,V)-Tupeln siehe unten).

Offensichtlich ist in einer Menge gesicherter (ω,V)-Antworttupel mit jedem Tupel t auch jedes Tupel enthalten, das aus t durch Ersetzung eines oder mehrerer Vorkommen von Konstanten oder Variablen durch ω entsteht. Die Definition paßt zur Definition gesicherter Antworttupel für ω-Zustände insofern, als diese Menge bei geeigneter Anpassung der Definition an V-Zustände stets in der Menge gesicherter (ω,V)-Antworttupel enthalten ist. Betrachte dazu die Definition

$$\mathcal{GT}(q,z_\omega) =_{df} \{t \mid t \text{ } \omega\text{-Tupel über } \beta \wedge (\forall z \in Z\text{-POSS}(z_\omega))(\exists t' \in q(z))(t' \in A\text{-POSS}(t))\}.$$

Angepaßt an V-Zustände ergibt sich aus dieser Definition:

$$\mathcal{GT}(q,z_V) =_{df} \{t \mid t \text{ } \omega\text{-Tupel über } \beta \wedge (\forall \nu\text{: } \nu \text{ Belegung für } \alpha, V) (\exists t' \in q(\nu(z_V)))(t' \in A\text{-POSS}(t))\}.$$

Da die Verwendung der Vervollständigung $\trianglerighteq$ der Verwendung der engen Ausdehnung als A-POSS-Funktion entspricht, kann die Menge gesicherter (ω-)Antworttupel demnach als Teilmenge der gesicherten (ω,V)-Tupel aufgefaßt werden.

Die für V-Tupel eingeführte genaue Überdeckungsrelation kann auf (ω,V)-Tupel wie folgt ausgedehnt werden:

Definition: Sei t ein (ω,V)-Tupel über einer Attributmenge $\alpha_i \subseteq \alpha$. Ein (ω,V)-Tupel t' über α_i ist eine genaue Überdeckung von t (in Zeichen: $t' \geq_g t$) $\Leftrightarrow_{df}$

1) $(\forall \nu$: ν Belegung für $\alpha, V \cup \{\omega\})(\exists \nu'$: ν' Belegung für $\alpha, V \cup \{\omega\})(\nu(t') = \nu'(t))$

2) Für alle $A \in \alpha_i$ mit t(A) ist definiert oder gleich einer Variablen aus V (d.h. ungleich ω) gilt:

t(A) = t'(A) oder

t(A) ist eine Variable und t'(A) ist definiert.

Beispiele: $(1,2,3) \geq_g (1,\omega_1,3)$, $(1,\omega_1,3) \geq_g (1,\omega,3)$, $(1,\omega_1,\omega) \geq_g (1,\omega,\omega)$.
Dagegen gilt nicht: $(1,\omega_1,3) \geq_g (1,\omega_2,\omega)$.

Eingeschränkt auf ω-Tupel oder V-Tupel erhalten wir die Definition der Überdeckungsrelation für ω-Tupel bzw. der genauen Überdeckungsrelation für V-Tupel.

Die Struktur der Menge aller gesicherten (ω,V)-Antworttupel ist mit der genauen Überdeckungsrelation wieder ein endlicher Infimum-Halbverband. Kleinstes Element ist das Tupel (ω,....,ω). Wegen dieser Eigenschaft können auch hier gesicherte Maximum- und min/max-Antworten definiert werden.

Als maximale Elemente können auch Tupel mit ω als Attributwert vorkommen. So erhalten wir für die zweite Anfrage in Beispiel 3.16 (S. 47) die beiden Tupel $(3,\omega_2)$ und $(2,\omega)$ als maximale Elemente.

Da sich die Definitionen dieser Antwortformen nicht wesentlich von denen für Anfragen in ω-Zuständen unterscheiden, sollen sie entfallen.

3.3 Mögliche Antworten

Mögliche Antworten haben wir oben als diejenigen Antworten definiert, die für mögliche Zustände erhalten werden. Da es im allgemeinen eine unendliche Anzahl möglicher Zustände gibt, kann auch die Anzahl möglicher Antworten unendlich sein. Eine Aufzählung möglicher Antworten ist aber auch im Fall einer endlichen Anzahl möglicher Antworten meist nicht sinnvoll. Dies liegt nicht nur daran, daß der Umfang der Ausgabe zu groß sein kann, sondern auch daran, daß Unterschiede zwischen den einzelnen Antworten durch einfaches Aufzählen der Antworttupel nur schwer auszumachen sind. Sinnvoller erscheint es daher, nach Formen für Antworten zu suchen, die den Schluß auf den Inhalt möglicher Antworten auf einfache Weise aus einer einzigen Relation heraus ermöglichen. Abstriche müssen dabei hinsichtlich der genauen Beschreibung möglicher Antworten gemacht werden. Abweichungen vom gewöhnlichen Tupeltyp können helfen, nicht zuviel Information zu verlieren.

Bemerkung: Eine andere Definition möglicher Antworten, die häufig betrachtet wird ([Lip79]), faßt in einer Antwort alle Tupel zusammen, die in möglichen Antworten enthalten sind. Das heißt, als mögliche Antwort wird die Vereinigung der Antworten definiert, die wir als mögliche Antworten bezeichnet haben. Hiermit ergibt sich die naheliegende Gegenüberstellung von gesicherter Antwort als Schnitt (untere Schranke) und möglicher Antwort als Vereinigung (obere Schranke) aller Tupel in Antworten zu möglichen Zuständen. Der Informationsverlust, der beim Übergang zur Vereinigung im allgemeinen eintritt, besteht darin, daß nicht mehr

bekannt ist, welche Tupel der Antwort miteinander in Antworten zu möglichen Zuständen auftreten dürfen. Dieses Problem stellt sich nicht, wenn, wie in ([Lip79]), nur Selektionen als Anfragen betrachtet werden.

Ein anderes Problem mit dieser Auffassung von möglichen Antworten ergibt sich dadurch, daß Antworten unendlich sein können, wenn im Typ der Antwort kein Schlüssel vorkommt, in dem die Antworttupel definiert sein müssen, d.h. wenn die Objektauffassung von [Lip79] nicht übertragbar ist. Es sind demnach auch hier praktikable Darstellungsformen anzugeben.

3.3.1 Gesicherte Antworten als Darstellungsformen für mögliche Antworten

Betrachten wir zunächst, wie sich die verschiedenen Formen gesicherter Antworten zur Darstellung möglicher Antworten eignen. Dabei beschränken wir uns auf Antworten zu Anfragen in ω-Zuständen. Für V-Zustände ergeben sich ähnliche Überlegungen.

Mit Ausnahme der gesicherten totalen Antwort beziehen sich die Definitionen aller Formen gesicherter Antworten auf die Menge gesicherter Antworttupel und können daher höchstens Information implizieren, die in einer solchen Menge schon enthalten ist. Ist eine Antwort in einem möglichen Zustand leer, dann ist auch die Menge gesicherter Antworttupel leer. In diesem Fall bietet es sich an, wie folgt vorzugehen: Leere Antworten werden bei der Bildung der Menge gesicherter Tupel nicht berücksichtigt; mit der Antwort wird die Information verknüpft, daß auch die leere Antwort eine mögliche Antwort ist. Mit dieser Festlegung können wir im folgenden annehmen, daß die Menge gesicherter Antworttupel stets nicht leer ist.

Offensichtlich kann unter der genannten Voraussetzung jede Antwort auf eine Anfrage in einem möglichen Zustand als enge Ausdehnung der Menge gesicherter Antworttupel erhalten werden. Das heißt, jede mögliche Antwort läßt sich erzeugen, indem alle partiellen Antworttupel durch eine oder mehrere geeignete Vervollständigungen ersetzt werden. Beim Übergang zur gesicherten totalen Antwort (die ebenfalls über die Menge gesicherter Antworttupel definiert werden kann) geht diese Eigenschaft im allgemeinen verloren. Sie ist nur dann gegeben, wenn es genau eine mögliche Antwort gibt.

Betrachten wir die gesicherte Maximum-Antwort. Die gesicherte Maximum-Antwort repräsentiert eindeutig die zugehörige Menge gesicherter Antworttupel. Da alle überdeckten Tupel aus dieser Menge aber fehlen, können mögliche Antworten im allgemeinen nicht als enge Ausdehnungen sondern nur als Ausdehnungen erhalten werden. Diese Antwortform liefert im allgemeinen eine bessere Beschreibung der Menge möglicher Antworten als die gesicherte totale Antwort, da sie auch partielle Tupel enthalten kann und damit die Menge der zugehörigen Ausdehnungen stets eine Teilmenge der Menge der Ausdehnungen für eine entsprechende gesicherte totale Antwort ist.

Nicht alle Überdeckungen von partiellen Tupeln in gesicherten Maximum-Antworten müssen in den zugehörigen möglichen Antworten vorkommen (siehe Beispiele 3.5 und 3.9, S. 40 bzw. 42). Gesicherte Maximum-Antworten sind daher wie folgt zu interpretieren:

Interpretation gesicherter Maximum-Antworten:

Sei q eine Anfrage an ein DB-Schema σ. Jede mögliche Antwort auf q in einem ω-Zustand z_ω zu σ ist eine Ausdehnung von $\mathcal{GA}_{max}(q,z_\omega)$. Es kann Vervoll-

ständigungen von $\mathcal{GA}_{max}(q,z_\omega)$ geben, die in keiner möglichen Antwort auf q in z_ω enthalten sind.

In der Definition gesicherter min/max-Antworten wird explizit gefordert, daß jede mögliche Antwort als mögliche Relation der Antwort (mit der gewählten A-POSS-Funktion) erhalten werden kann. Wie wir gesehen haben, kommt als POSS-Funktion für diese Antwortform nur die enge Ausdehnung in Betracht. Bezüglich der Interpretation der Vorkommen von ω in Antworttupeln gilt das gleiche wie für die gesicherte Maximum-Antwort. Gesicherte min/max-Antworten sind daher wie folgt zu interpretieren:

Interpretation gesicherter min/max-Antworten:

Sei q eine Anfrage an ein DB-Schema σ, und sei T eine gesicherte min/max-Antwort auf q in einem ω-Zustand z_ω zu σ.
Jede mögliche Antwort auf q in z_ω ist eine mögliche Relation zu T (mit $POSS_{ce}$ als Möglichkeitsfunktion). Es kann Vervollständigungen von T geben, die in keiner möglichen Antwort zu q in z_ω enthalten sind.

Will man die Güte von Antworten hinsichtlich der Beschreibung möglicher Antworten messen, bietet es sich an, die Mengen möglicher Relationen zu den Antworten herzunehmen und die Mengeninklusion als Ordnungskriterium festzulegen. Zusätzlich kann bei Gleichheit der Mengen möglicher Relationen die Minimalität der Darstellung als Ordnungskriterium hinzukommen. Ähnliche Festlegungen finden sich auch bei intensionalen Antworten auf Anfragen (siehe etwa [Mot94]).

Für die gesicherten totalen, Maximum- und min/max-Antworten gilt offensichtlich für alle Anfragen q und ω-Zustände z_ω:

$$\mathcal{GA}_{total}(q,z_\omega) \subseteq \mathcal{GA}_{max}(q,z_\omega) \subseteq \mathcal{GA}_{min/max}(q,z_\omega).$$

Wegen der umgekehrten Monotonieeigenschaft der POSS-Funktionen und da jede enge Ausdehnung einer ω-Relation R auch eine Ausdehnung von R ist, folgt:

$$POSS_{ce}(\mathcal{GA}_{min/max}(q,z_\omega)) \subseteq POSS_e(\mathcal{GA}_{max}(q,z_\omega)) \subseteq POSS_e(\mathcal{GA}_{total}(q,z_\omega)).$$

Dabei steht $POSS_e$ für die Ausdehnung als Möglichkeitsfunktion.

Falls die betrachteten Antwortformen für eine Anfrage zu unterschiedlichen Ergebnissen führen, erhalten wir demnach die beste Beschreibung möglicher Antworten mit einer gesicherten min/max-Antwort.

Die beiden Formen gesicherter uni-Antworten führen in der Regel zu Antworten, die sich bezüglich der Mengeninklusion nur untereinander sowie mit gesicherten totalen Antworten vergleichen lassen. Dies zeigt Beispiel 3.13 (S. 45). Aus den Definitionen der beiden Antwortformen folgt unmittelbar, daß die strenge gesicherte uni-Antwort stets in der gesicherten uni-Antwort enthalten ist. Als Interpretationen ergeben sich:

Interpretation strenger gesicherter uni-Antworten:

Sei q eine Anfrage an ein DB-Schema σ im Zustand z_ω. Jede mögliche Antwort zu q in z_ω ist eine Ausdehnung von $\mathcal{GA}_{s\text{-}uni}(q,z_\omega)$. Jede Vervollständigung eines Tupels von $\mathcal{GA}_{s\text{-}uni}(q,z_\omega)$ ist in einer möglichen Antwort auf q in z_ω enthalten.

Interpretation gesicherter uni-Antworten:

Sei q eine Anfrage an ein DB-Schema σ im Zustand z_ω. Jede mögliche Antwort zu q in z_ω ist eine Ausdehnung von $\mathcal{GA}_{uni}(q,z_\omega)$. Jede Vervollständigung eines Tupels

von $\mathcal{GA}_{uni}(q,z_\omega)$ ist in einer möglichen Antwort auf q in z_ω enthalten, oder es kann durch Ersetzung einzelner Werte durch andere Werte des Tupels ein Tupel erhalten werden, das in einer möglichen Antwort auf q in z_ω enthalten ist.

3.3.2 Weitere Formen zur Darstellung möglicher Antworten

In [Bis83] werden partitionierte ω-Relationen betrachtet, wobei zwischen Tupeln, die mit Sicherheit zu einer Relation gehören, und Tupeln, die möglicherweise zu einer Relation gehören, unterschieden wird (siehe hierzu Kapitel 5). In diesem Sinn könnten den gesicherten Antworten ebenfalls Tupel hinzugefügt werden, die entsprechend als mögliche Tupel markiert sind. Hierdurch ließe sich in vielen Fällen die Ausdehnung durch die enge Ausdehnung oder die Vervollständigung als A-POSS-Funktion ersetzen.

In Abhängigkeit vom betrachteten DB-Zustand kann es auch sinnvoll sein, statt Variablen oder dem Symbol ω Mengen von Konstanten oder Boolesche Ausdrücke als Attributwerte zuzulassen und Regeln anzugeben, wie aus Antworten mit derart verallgemeinerten Tupeln mögliche Antworten gebildet werden können. Darauf haben wir bereits in den Beispielen 3.6 und 3.7 (S. 41) hingewiesen.

4 Vervollständigungen von ω-Relationen

Im folgenden wollen wir zunächst zwei Ansätze aus der Literatur betrachten, die sich mit der Auswertung von Anfragen an partielle Datenbankzustände beschäftigen. In beiden Ansätzen bedeutet das Fehlen eines Attributwertes aus dem Wertebereich, daß genau ein Wert für das entsprechende Tupel und Attribut existiert, dieser Wert aber nicht bekannt ist. Gemeinsam ist beiden Vorschlägen auch, daß ein einziges Symbol zur Repräsentation fehlender Werte benutzt wird. Die betrachteten partiellen Zustände entsprechen damit ω-Zuständen mit der Vervollständigung als POSS-Funktion.

Im Anschluß an die Diskussion der beiden Ansätze geben wir mehrere Verfahren zum Auswerten von TRC- und Algebraanfragen an, durch die Teilmengen verschiedener Formen gesicherter Antworten erhalten werden können, und zeigen die (klassische) Äquivalenz von TRC und Relationenalgebra für eine Interpretationsvorschrift und ein Auswertungsverfahren.

4.1 Der Vorschlag von E.F. Codd

Eine Grundforderung, die E.F. Codd schon bei der Einführung des relationalen Datenmodells ([Cod70], [Cod72a]) aufgestellt hat und die er in seinen Beiträgen zur Nullwertproblematik ([Cod75], [Cod79], [Cod86], [Cod87], [Cod90]) wiederholte, ist die Definiertheit aller Werte in den Attributen des Primärschlüssels. Mit dieser Annahme steht jedes Vorkommen eines Nullwertes in den Relationen eines Datenbankzustands für genau einen fehlenden Wert. In Kapitel 2 haben wir schon angesprochen, welche Auswirkungen die Anwendung von Operationen auf die Interpretation der Vorkommen von Nullwerten haben können, wenn die Mengeneigenschaft von Relationen beibehalten wird. In [Cod79] wird die Mengeneigenschaft von Relationen beim Auswerten von Ausdrücken durch eine *Duplikatentfernungsregel* sichergestellt. Diese Regel besagt, daß zwei Tupel als identisch anzusehen sind, wenn sie in allen definierten Werten übereinstimmen. Dabei spielt es keine Rolle, woher die Werte der Tupel stammen, d.h. ob etwa zwei Nullwerte in einem Vergleich auf dasselbe Vorkommen im Zustand zurückgehen. Für die Auswertung von Ausdrücken, denen Wahrheitswerte zuzuordnen sind (Boolesche Ausdrücke, Mengeninklusionen), wird dagegen das sogenannte *Nullersetzungsprinzip* (*null substitution principle*) angewandt. Gemäß diesem Prinzip ist einem Ausdruck α, in dem alle Variablen durch definierte Werte oder den Nullwert ω ersetzt worden sind, wie folgt einer der Werte *wahr*, *falsch* oder *unbekannt* zuzuordnen:

Nullersetzungsprinzip

α hat den Wert *unbekannt* genau dann, wenn beide der folgenden Bedingungen gelten:

1) Jedes Vorkommen eines unbekannten Wertes in α kann durch einen definierten Wert derart ersetzt werden, daß der resultierende Ausdruck bei Auswertung gemäß der Booleschen Logik den Wert *wahr* hat; dabei darf jedes dieser Vorkommen unabhängig von den anderen Vorkommen ersetzt werden.
2) Jedes Vorkommen eines unbekannten Wertes in α kann durch einen definierten Wert derart ersetzt werden, daß der resultierende Ausdruck bei Auswertung

gemäß der Booleschen Logik den Wert *falsch* hat; dabei darf jedes dieser Vorkommen unabhängig von den anderen Vorkommen ersetzt werden.

In allen anderen Fällen hat α entsprechend den Wert *wahr* oder *falsch*.

Für die Auswertung der Booleschen Ausdrücke mit den drei Wahrheitswerten wird eine dreiwertige Logik benutzt. Es werden nur die Junktoren $\vee$, $\wedge$ und $\neg$ betrachtet und hierfür Definitionen angegeben, die mit den Festlegungen der von J. Lukasiewicz (1921) und S.C. Kleene (1938) eingeführten Logiken übereinstimmen ([Kle38], [Kle52]; zur Logik von Lukasiewicz siehe zum Beispiel [Res69] oder [BB94]). Die Auswertungsvorschrift für diese Logik ergibt sich aus den folgenden Tabellen für die Junktoren $\vee$, $\wedge$ und $\neg$:

$\vee$	W	F	U
W	W	W	W
F	W	F	U
U	W	U	U

$\wedge$	W	F	U
W	W	F	U
F	F	F	F
U	U	F	U

$\neg$	
W	F
F	W
U	U

Eine wesentliche Eigenschaft dieser Logik, die sich direkt aus den Tabellen ablesen läßt, ist die sogenannte Monotonieeigenschaft (vergleiche [DT91]):

Monotonieeigenschaft

Wird in einem beliebigen Ausdruck, der den Wert *wahr* (*falsch*) hat, ein Vorkommen des Wertes ω durch *wahr* oder *falsch* ersetzt, dann ändert sich der Wert des Ausdrucks nicht, er bleibt *wahr* (*falsch*).

Bemerkung: Die von Lukasiewicz und Kleene definierten Logiken (Kleene hat noch eine weitere dreiwertige Logik angegeben, die hier aber nicht von Interesse ist) unterscheiden sich in der Definition der Implikation und damit auch der Äquivalenz. Während in der Logik von Lukasiewicz für U $\Rightarrow$ U der Wert *wahr* festgelegt ist, sind in der Kleene'schen Logik alle von $\vee$, $\wedge$ und $\neg$ verschiedenen Junktoren gemäß den Äquivalenzen der zweiwertigen Logik definiert.

Auch wenn in einer Sprache wie SQL nur die Junktoren $\vee$, $\wedge$ und $\neg$ vorkommen, ist es für Äquvalenz erhaltende Transformationen, wie sie etwa bei der Optimierung von Anfragen benötigt werden, wichtig, welche der beiden Logiken verwendet wird. So gilt etwa nur in der Kleene'schen Logik die Äquvalenz von A $\wedge \neg$ B $\Rightarrow \neg$ C und C $\wedge \neg$ B $\Rightarrow \neg$ A. Wir werden darauf in Kapitel 7 zurückkommen.

Als Begründung für die unterschiedliche Behandlung unbekannter Attributwerte in der Duplikatentfernungsregel (alle Vorkommen von ω werden identifiziert) und beim Nullersetzungsprinzip ($\omega = \omega$ erhält den Wert *unbekannt*) wird in [Cod86] angeführt, daß die Elimination von Duplikaten in Relationen zu keinerlei Problemen führt, "because the semantic notions of equality are applicable at a higher level of abstraction than the symbolic equality involved in removal of duplicate tuples" ([Cod86], S. 67).

Als Folge dieser unterschiedlichen Behandlung kann es zu unerwünschten Inäquivalenzen von Ausdrücken kommen, die im Fall des Fehlens unbekannter Werte für alle Zustände äquivalent sind. Zum Beispiel liefert der Schnitt zweier Relationen im allgemeinen nicht mehr das gleiche Ergebnis wie ihr Join. Daß nicht alle Äquivalenzen, die für mengentheoretische und Boolesche Ausdrücke gelten, bei Verwendung

der dreiwertigen Logik gelten können, ist leicht einzusehen. Sinnvoll erscheint jedoch, diese Inäquivalenzen so weit wie möglich zu vermeiden. Eine weitere und schwerwiegendere Konsequenz der von Codd vertretenen Auffassung ist, daß die in [Cod79] und [Cod86] diskutierten TRUE- und MAYBE-Versionen von Antworten auf Anfragen im allgemeinen nicht als gesicherte bzw. mögliche Antworten angesehen werden dürfen. Betrachten wir zunächst Beispiele für Probleme mit TRUE-Versionen von Antworten.

Beispiel 4.1: Seien R = {(1,ω), (2,1)} und S = {(1,ω), (2,1), (3,2)} ω-Relationen über {A,B}.

Für R \ S erhalten wir die leere Relation über {A,B} als Ergebnis, da die beiden Vorkommen von ω in R und S gemäß der Duplikateliminationsregel identifiziert werden. Damit ergibt sich für den Ausdruck R \ (R \ S) mit dem Funktionalitätsprinzip der Relationenalgebra die Relation R als Resultat. Dieses Ergebnis stellt keine gesicherte Information dar, da etwa mit der Ersetzung des Vorkommens von ω in R durch 3 und desjenigen in S durch 4 als Ergebnis {(2,1)} erhalten wird. R als Resultat muß somit als mögliche Antwort interpretiert werden.

Für den Ausdruck R ⊆ R erhalten wir andererseits gemäß dem Nullersetzungsprinzip *unknown* als Resultat, da die Ersetzung von ω in den beiden Operanden unabhängig voneinander erfolgt.

Die vorgeschlagene Methode kann sogar dazu führen, daß eine Antwort auf eine Anfrage zu keiner möglichen Antwort paßt. Dies ist sowohl für die TRUE-Versionen als auch für die MAYBE-Versionen der von Codd angegebenen erweiterten Algebraoperationen möglich. Betrachten wir zunächst ein Beispiel mit TRUE-Versionen von Operationen.

Beispiel 4.2: Seien T = {(2,1)} und U = {(2,ω)} ω-Relationen über {A,B}. Als Ergebnis des Ausdrucks T[A] \ (U[B=1] ∪ U[B≠1])[A] erhalten wir {(2)}, da die Tautologie mit dem Nullersetzungsprinzip nicht erkannt wird. Der Gesamtausdruck liefert aber für alle möglichen Relationen zu U eine leere Relation als Antwort.

In den MAYBE-Versionen werden in das Ergebnis einer Θ-Join- oder Selektions-Operation genau diejenigen Tupel aufgenommen, für die bei der Auswertung der zugehörigen Bedingung *unknown* erhalten wird. Entsprechend wird bei der Division verfahren. Zur Kennzeichnung erhalten die Operationen den Index ω.

Beispiel 4.3: Seien V = {(3,1), (3,2)} und W = {(1,ω)} ω-Relationen über {A,B} bzw. {C,D}.
Betrachte folgenden Ausdruck: (V[B $=_\omega$ D]W)[A,B]. Als Ergebnis der MAYBE-Version des Θ-Join erhalten wir {(3,1,ω), (3,2,ω)} und somit nach der anschließenden Projektion wieder V. Unter der Annahme, daß jedes Vorkommen von ω genau einen unbekannten Wert repräsentiert, kann dieses Ergebnis für keine mögliche Relation zu W erhalten werden. Die Antwort muß in diesem Fall so interpretiert werden, daß die in ihr enthaltenen Tupel (3,1) und (3,2) zwar in möglichen Antworten vorkommen, nicht jedoch in der gleichen Antwort.

Die Problematik der Duplikateliminationsregel im Zusammenhang mit der Vervollständigung als POSS-Funktion wurde neben anderen Fragestellungen von J. Biskup in [Bis83] behandelt. Da dort die enge Ausdehnung statt der Vervollständigung als POSS-Funktion betrachtet wird, um die aufgezeigten Probleme zu vermeiden, beschäftigen wir uns mit den Vorschlägen von Biskup im nächsten Kapitel.

4.2 Die Untersuchungen von T. Imielinski und W. Lipski

In [IL84] wird untersucht, ob und wie Operationen der Relationenalgebra auf verschiedene Formen von Relationen mit unvollständiger Information erweitert werden können, wenn an die Ergebnisse gewisse Bedingungen bezüglich ihrer Darstellbarkeit gestellt werden. Neben ω-Relationen, in [IL84] *Codd-Relationen* genannt, werden dabei auch V-Relationen betrachtet, auf die wir in Kapitel 6 näher eingehen. Die Untersuchungen zu ω-Relationen erfolgen unter der OWA mit der Ausdehnung als POSS-Funktion. Da die erhaltenen Ergebnisse aber auch unter der CWA mit der Vervollständigung als POSS-Funktion gelten (Satz 9.1 in [IL84]), wollen wir sie in diesem Kapitel diskutieren.

Bemerkung: In [IL84] werden für den Übergang von der OWA zur CWA ω-Zustände in V-Zustände überführt, indem jedes Vorkommen von ω in einem ω-Zustand durch eine eindeutige Variable ersetzt wird. Derartige Überführungen sind im allgemeinen nicht informationserhaltend, wenn für ω-Zustände die Ausdehnung und für V-Zustände Belegungen als POSS-Funktionen genommen werden. Dies haben wir in Kapitel 2 im Zusammenhang mit update-Operationen diskutiert. Die in [IL84] gewählte Vorgehensweise führt aber nicht zu Schwierigkeiten, da nur Resultate formuliert werden, die auf einem Äquivalenzbegriff aufbauen, der über Schnittmengen definiert ist, in denen nur totale Tupel enthalten sind.

Die oben erwähnten Bedingungen an Auswertungsergebnisse sind für sogenannte Multitabellen angegeben, die DB-Zuständen entsprechen. Statt Ausdrücken werden Tupel von Ausdrücken betrachtet, die auf Multitabellen wirken und neue Multitabellen erzeugen. Dadurch werden Aussagen bezüglich der Funktionalitätseigenschaft von Operationen in Ausdrücken der Relationenalgebra gut formulierbar. Um zu einem Verständnis der Vorgehensweise zu gelangen, genügt es aber, bei einfachen Ausdrücken zu bleiben und diese weiterhin als Abbildungen von DB-Zuständen in DB-Relationen anzusehen.

Für eine feste Zustandsmenge mit Möglichkeitsfunktion POSS und eine ebenfalls feste Teilmenge der Operationen der Relationenalgebra wird untersucht, ob es für jeden Zustand möglich ist, das Ergebnis eines Ausdrucks unter Verwendung der gleichen Funktion POSS so darzustellen, daß bezüglich gesicherter totaler Antworten (!) für alle mit den betrachteten Operatoren bildbaren Ausdrücke kein Informationsverlust in folgendem Sinn auftritt:

Seien z_ω ein ω-Zustand zu einem DB-Schema σ und $e_1,\ldots,e_k$ beliebige Ausdrücke zu σ, die mit den Operatoren der gegebenen Teilmenge $\mathfrak{O}$ der Algebraoperatoren gebildet werden können; sei M die Menge der möglichen Zustände, die aus den möglichen Zuständen zu z_ω durch Hinzufügen der jeweiligen möglichen Antworten für $e_1,\ldots,e_k$ erhalten werden. Sei ferner $e_i(z_\omega)$, $i = 1,\ldots,k$ das Ergebnis der Auswertung von e_i in z_ω. Dann soll für jeden Ausdruck b, der mit den Operatoren aus $\mathfrak{O}$ gebildet werden kann, die gesicherte totale Antwort auf b in $z_\omega \cup \bigcup_{i=1}^{k} e_i(z_\omega)$ gleich sein dem Schnitt aller Antworten auf b in den möglichen Zuständen aus M, d.h. es soll gelten:

$$(*) \quad \bigcap \{b(m) \mid m \in M\} = \bigcap \{b(z) \mid z \in \mathrm{POSS}(z_\omega \cup \bigcup_{i=1}^{k} e_i(z_\omega))\}.$$

Es wird demnach (nur) verlangt, daß die Antworten zu den e_i in z_ω die möglichen Antworten zu den e_i in z_ω so repräsentieren, daß bezüglich gesicherter totaler

Antworten auf beliebige Anfragen mit den Operatoren aus $\mathfrak{O}$ keine Information verlorengeht. Ist eine solche Eigenschaft für eine gewählte Darstellungsform $\mathfrak{T}$ partieller DB-Relationen in Zuständen mit Möglichkeitsfunktion POSS und eine feste Operatorenmenge $\mathfrak{O}$ (mit Festlegung der Auswertung in ω-Zuständen) vorhanden, dann wird ($\mathfrak{T}$, POSS, $\mathfrak{O}$) ein *Repräsentantensystem* genannt.

In [IL84] wird gezeigt, daß es Repräsentantensysteme für ω-Relationen nur mit der leeren Operatorenmenge sowie mit Operatorenmengen gibt, in denen nur die Projektion und/oder die Selektion (mit Beschränkung der Operatoren in Vergleichsausdrücken auf die Gleichheit) vorkommen.

Beispiel 4.4: Sei R = {(1,2,ω), (2,2,ω), (ω,2,3)} eine Relation über {A,B,C}. $\mathfrak{O}$ bestehe aus allen passenden Projektionen und Selektionen. Betrachte den Ausdruck R[B,C]. Mit der üblichen Definition von Projektionen übertragen auf ω-Relationen erhalten wir {(2,ω), (2,3)} als Ergebnis. Für diese Relation ergeben sich mit $POSS_{ce}$ genau alle möglichen Antworten als mögliche Relationen. Damit ist keine Information verlorengegangen.

Betrachte den Ausdruck (R[A=2])[B,C]. Als Antwort können wir hier {(2,ω)} festlegen, ein Ergebnis, das etwa mit den in Abschnitt 4.1 betrachteten TRUE-Versionen der Algebraoperationen erhalten wird. Das Tupel (2,3) muß in der Antwort fehlen, da es nicht in jeder möglichen Antwort enthalten ist. Damit kann es aber auch nicht zum Schnitt aller Antworten auf Anfragen an diejenigen (totalen) Zustände beitragen, die den möglichen Antworten für den Ausdruck entsprechen.

Betrachte einen Ausdruck, in dem die Vereinigung als Operation vorkommt:
(R[C=3] ∪ R[C≠3])[A].
Für die beiden Operanden R[C=3] und R[C≠3] kommen als Darstellungen der möglichen Antworten nur {(ω,2,3)} bzw. die leere DB-Relation über {A,B,C} in Frage. Für den gesamten Ausdruck ist die gesicherte totale Antwort dagegen {(1), (2)}, d.h. aus den möglichen Darstellungen der Operandenergebnisse nicht zu erhalten. Damit kann die Vereinigung in der Operatorenmenge eines Repräsentantensystems für ω-Relationen mit $POSS_{ce}$ als Möglichkeitsfunktion nicht zusammen mit Selektionen und Projektionen vorkommen.

Die Bezugnahme auf gesicherte totale Antworten führt zu einer Schwäche des Begriffs Repräsentantensystem. Betrachte den Ausdruck R[C=3] ∪ R[C≠3], d.h. den letzten Ausdruck aus Beispiel 4.4 ohne die Projektion auf das Attribut A. Für diesen Ausdruck ist die gesicherte totale Antwort die leere DB-Relation über {A,B,C}. Wir erhalten also kein Beispiel, das die für Repräsentantensysteme geforderte Eigenschaft (*) verletzt. Es läßt sich einfach einsehen, daß mit der Selektion und der Vereinigung allein diese Eigenschaft mit ω-Relationen und der Vervollständigung als POSS-Funktion auch nicht verletzt werden kann. Offensichtlich gilt aber R[C=3] ∪ R[C≠3] = R.

Es erscheint für die Definition von Repräsentantensystemen sinnvoller zu sein, auf die Menge gesicherter Antworttupel Bezug zu nehmen anstatt auf totale gesicherte Antworten. Ist die Projektion in der Operandenmenge enthalten, kann der Bezug auf totale gesicherte Antworten dagegen beibehalten werden: Durch geeignete Erweiterung von Relationen um ein Attribut, in dem nur totale Werte vorkommen, kann aus jedem Beispiel für eine Verletzung der Eigenschaft eines Repräsentantensystems mit Bezug auf die Menge gesicherter Antworttupel ein Beispiel für das entsprechende ursprüngliche Repräsentantensystem konstruiert werden.

4.3 Berechnung gesicherter Antworten für TRC-Anfragen

Bei den Definitionen der verschiedenen Formen gesicherter Antworten haben wir zwischen POSS-Funktionen für DB-Zustände und POSS-Funktionen für Antworten unterschieden. Wie die beim Vorschlag von E.F. Codd auftretenden Probleme zeigen, ist es im allgemeinen nicht sinnvoll, für gesicherte Antworten die Vervollständigung als POSS-Funktion zu wählen, wenn Antworten Tupelmengen sein sollen, d.h. keine Duplikate enthalten dürfen. Eine einfache Projektion kann hier schon zu falschen Deutungen der in Antworten enthaltenen Information führen.

Läßt man in Antworten Duplikate zu und legt man als POSS-Funktion die Vervollständigung zugrunde, dann gibt die Anzahl der Duplikate eines ω-Tupels in einer gesicherten Antwort an, wieviele verschiedene totale Tupel in einer möglichen Antwort von diesen Duplikaten höchstens repräsentiert werden. Vorausgesetzt ist dabei natürlich eine geeignete Auswertungsstrategie. Wir beschäftigen uns hier nicht mit solchen Antworten, sondern betrachten nur gesicherte Antworten, wie sie in Kapitel 3 definiert wurden. Als POSS-Funktion für Antworten wählen wir die enge Ausdehnung, um die angesprochenen Interpretationsprobleme zu vermeiden.

4.3.1 Die ω-Auswertung

In der üblichen modelltheoretischen Betrachtungsweise des relationalen Datenmodells wird die Semantik einer Kalkülsprache wie in der Prädikatenlogik durch Angabe einer Vorschrift festgelegt, die bestimmt, wann ein Ausdruck α der Sprache unter (bei) einer Interpretation I gilt. Man sagt dann auch: I erfüllt α. Hierbei wird ein Zustand als fest vorgegeben angenommen, so daß eine Interpretation schon durch die Angabe einer Belegung für die freien Variablen einer Anfrage bestimmt ist. Mit welchen Tupeln Variable im TRC zu belegen sind, hängt davon ab, welche Variante des TRC zugrunde liegt. Für eine ausführliche Diskussion verschiedener Varianten sei auf [KK93] verwiesen. Wir wollen im folgenden erlaubte TRC-Anfragen betrachten, wie sie im vorangehenden Kapitel definiert wurden. Die Gründe hierfür liegen darin, daß diese Sprachklasse im Fall totaler Zustände die gleiche Ausdruckskraft hat wie die Relationenalgebra (mit konstanten Relationen, aber ohne Komplement) und daß erlaubte Anfragen sich einfacher als Anfragen anderer Sprachklassen mit gleicher Ausdruckskraft in Ausdrücke der Relationenalgebra übersetzen lassen und umgekehrt. Dies ermöglicht uns geeignete Vergleiche der im folgenden vorgeschlagenen Interpretationsvorschriften und erweiterten Algebraoperationen für partielle Zustände.

Für erlaubte Anfragen genügt es bei totalen Zuständen, die Variablen mit Tupeln aus dem beschränkten Universum zu belegen, d.h. mit Tupeln, deren Werte entweder im Zustand oder in der Anfrage vorkommen (man spricht dann auch von *beschränkter Interpretation*). Die Belegungsmöglichkeiten können noch weiter eingeschränkt werden, ohne daß sich die Antworten ändern: Es genügt, alle Variablen mit Vorkommen, die über Bereichsausdrücke beschränkt sind (*bereichsbeschränkte Vorkommen*) mit Tupeln aus den zugehörigen Relationen des betrachteten Zustands zu belegen und für Variable mit Vorkommen, die nur über Vergleichsausdrücke beschränkt sind (*vergleichsbeschränkte Vorkommen*), zusätzlich die entsprechenden Konstanten oder Werte aus den Tupeln, durch die sie direkt oder indirekt beschränkt sind, für die Bildung von Belegungen herzunehmen.

Als Beispiel betrachte die Anfrage

$(x)/\ x.A = 3 \ \vee \ (\exists y)(R\ y \ \wedge \ x.A = y.B)$.

Hier sind als Belegungen für y die Tupel aus der Relation R im betrachteten Zustand zu nehmen und für x die Projektionen der Belegungen für y auf B sowie zusätzlich das Tupel (ω), da x im Disjunktionsglied x.A = 3 nicht über y oder über einen Bereichsausdruck beschränkt ist.

Wird ein Attribut A des Typs einer Variablen x in einem vergleichsbeschränkten Vorkommen von x an ein Attribut B einer Existenz-quantifizierten Variablen y gebunden (wie z.B. A von typ(x) an B von typ(y) in $(x)/\ (\exists y)(R\ y \wedge x.A = y.B)$), so ergibt sich aus jeder Belegung t der Existenz-quantifizierten Variablen y eine Belegung für die Variable x mit dem entsprechenden Attributwert aus t für den Wert unter A. Eine derartige Bindung an ein Attribut einer universell quantifizierten Variablen kann in einem erlaubten Ausdruck nicht erfolgen, da jede universell gebundene Variable im Wirkungsbereich des Quantors negativ beschränkt sein muß.

Die positive Beschränktheit aller freien und Existenz-quantifizierten Variablen sowie die negative Beschränktheit aller universell quantifizierten Variablen in erlaubten Anfragen garantiert, daß andere Belegungen als die angesprochenen überflüssig sind für die Auswertung: Für freie Variable und Existenz-quantifizierte Variable ergibt sich bei solchen Belegungen der Wert *falsch* für den Qualifikationsausdruck der Anfrage bzw. den quantifizierten Teilausdruck; für universell quantifizierte Variable erhalten wir den Wert *wahr*, d.h. ebenfalls das "neutrale Element" für den Wirkungsbereich des Quantors.

Betrachten wir nun Möglichkeiten zur Auswertung erlaubter TRC-Ausdrücke über ω-Zuständen. Variable müssen hier im allgemeinen mit ω-Tupeln belegt werden können, um in Interpretationen die Zuordnung zu ω-Tupeln in ω-Relationen des Zustands zu ermöglichen. Das unbeschränkte und das beschränkte Universum sind daher stets um ω-Tupel geeignet zu erweitern. Zur Bewahrung der Idee, die der Definition erlaubter Ausdrücke zugrunde liegt, sollte unter dieser Erweiterung eine Auswertung bei unbeschränktem Universum das gleiche Ergebnis liefern wie eine Auswertung bei beschränktem Universum.

Eine Schwierigkeit mit dem Wert ω in Belegungen ist dadurch gegeben, daß verschiedene Vorkommen dieses Wertes nicht automatisch wie gleiche definierte Werte behandelt werden dürfen. Es ist vielmehr darauf zu achten, daß unzulässige Identifizierungen vermieden werden. Die oben angegebene Möglichkeit zur Beschränkung von Belegungen auf diejenigen Tupel, die in den Relationen des betrachteten Zustands enthalten sind oder die sich aus vergleichsbeschränkten Vorkommen von Variablen ergeben, kann zum Verlust gesicherter Information in Antworten führen: Gesicherte Antworttupel müssen nicht in der Relation eines Zustands vorkommen, auch wenn die zugehörige Variable über einen Bereichsausdruck positiv beschränkt ist. Dies haben wir schon bei der Diskussion der verschiedenen Formen gesicherter Antworten gesehen. Entsprechend kann ein Verlust eintreten, wenn für die Belegung gebundener Variablen nur Tupel des Zustands in der beschriebenen Weise berücksichtigt werden.

Um zu einer einfachen Interpretationsvorschrift zu gelangen, betrachten wir im folgenden zwar Belegungen mit Tupeln aus dem mit ω-Tupeln erweiterten beschränkten Universum, gestalten die Vorschrift aber so, daß eine Beschränkung auf Belegungen in der oben genannten Weise zu gleichen Antworten führen würde.

Da die Anzahl möglicher Belegungen von Variablen wegen der Endlichkeit des beschränkten Universums stets endlich ist, können quantifizierte Teilausdrücke bei der Auswertung von TRC-Ausdrücken durch endliche Disjunktionen ($\exists$) bzw. endliche Konjunktionen ($\forall$) äquivalent ersetzt werden. Es genügt für die Interpretationsvorschrift daher festzulegen, wie Bereichs- und Vergleichsausdrücken Werte zuzuordnen und wie Boolesche Ausdrücke über diesen Ausdrücken auszuwerten sind.

Betrachten wir zunächst Bereichsausdrücke. Es ist naheliegend, einem Ausdruck R x unter einer Interpretation I wie bei gewöhnlichen Interpretationen den Wert *wahr* zuzuordnen, wenn das Tupel I(x) in der Relation I(R) enthalten ist. Diese naive Vorgehensweise kann aber zu nicht gesicherten Antworten führen, wie das folgende Beispiel zeigt.

Beispiel 4.5: Seien R und S ω-Relationen über {A,B}, die beide $(1,\omega)$ als einziges Tupel enthalten. Betrachte folgende erlaubte TRC-Anfrage:

$(x)/\ R\ x\ \wedge\ S\ x$

Als Antwort auf diese Anfrage erhält man $\{(1,\omega)\}$, wenn beide Bereichsausdrücke zu *wahr* ausgewertet werden für die einzige Belegungsmöglichkeit $(1,\omega)$ von x. Wählen wir z.B. $\{(1,2)\}$ und $\{(1,3)\}$ als mögliche Relationen zu R bzw. S, dann ergibt sich mit der gewöhnlichen Interpretationsvorschrift aber eine leere Antwort. Die Menge gesicherter Antworttupel ist demnach ebenfalls leer.

Wenn Anfragen mit derartigen Identifizierungsmöglichkeiten für Vorkommen der gleichen Belegung nicht durch Bedingungen an die Syntax von Anfragen ausgeschlossen werden sollen, müssen nicht zulässige Identifizierungen durch die Interpretationsvorschrift ausgeschlossen werden. Um dies zu erreichen, gehen wir wie folgt vor:

Bei jeder Interpretation wird für Bereichsausdrücke und beschränkende Vergleichsausdrücke zwischen *definierenden* und *gewöhnlichen* Ausdrücken unterschieden. Beschränkende Vergleichsausdrücke sind dabei Vergleichsausdrücke, durch die das Vorkommen einer vergleichsbeschränkten Variablen an Konstanten oder Werte anderer Tupel gebunden wird (siehe Definition erlaubter Ausdrücke in Kapitel 3). Jede Variablenbelegung bestimmt dadurch im allgemeinen nicht eine einzelne Interpretation sondern eine Menge von Interpretationen gemäß den Möglichkeiten, die sich aus folgendem Vorgehen ergeben:

Sei α ein erlaubter TRC-Ausdruck. Zu jeder Variablen x in α werden von den Bereichs- bzw. Vergleichsausdrücken einige zu definierenden Ausdrücke bestimmt, so daß gilt:

- α bleibt erlaubter Ausdruck, auch wenn bei der Überprüfung dieser Eigenschaft nur die als definierend markierten Ausdrücke für die Beschränkung von x berücksichtigt werden;
- wird ein einziger dieser Ausdrücke dagegen nicht berücksichtigt, dann verliert α die Eigenschaft "erlaubt".

Beispiel 4.6: a) Für den TRC-Ausdruck $R\ x \wedge S\ x$ muß entweder R x oder S x als definierend markiert werden. Wegen der geforderten Minimalität können nicht beide Bereichsausdrücke in einer Interpretation definierend sein.

b) Für $R\ x \vee S\ x$ gibt es nur eine Möglichkeit: Beide Bereichsausdrücke müssen definierend sein, damit der gesamte Ausdruck erlaubt ist.

c) Für $R\ x \wedge \neg S\ x$ kann nur R x als definierend markiert sein.

d) Bei U x $\wedge$ $\neg$ ($\exists$y)(R y $\wedge$ S y) muß U x stets definierend sein; für R y $\wedge$ S y gilt das gleiche wie in a).

e) Für U x $\wedge$ ($\forall$y)($\neg$ R y $\vee$ S y) gibt es nur eine Möglichkeit: U x und R y müssen definierend sein, S y muß gewöhnlich sein.

Zu jeder Interpretation gehört somit für jede Variable neben der Belegung auch die Angabe, welche Bereichsausdrücke bzw. welche Vergleichsausdrücke als definierend anzusehen sind. Für die Beispielanfrage (x)/ R x $\wedge$ S x ergeben sich daher aus der Belegung von x mit (1,ω) zwei Interpretationen, wobei einmal R x und das andere Mal S x definierender Bereichsausdruck ist.

Für die Anfrage (x)/ (R x $\vee$ S x) $\wedge$ x.A > 3 ergibt sich aus der gleichen Belegung dagegen nur eine Interpretation, da R x und S x definierend sein müssen, damit die Anfrage erlaubt ist.

Die Idee hinter dieser Festlegung ist folgende:

Jedes Belegungstupel wird eindeutig einem Tupel einer Relation (definierender Bereichsausdruck) oder Konstanten und/oder Werten anderer Belegungstupel (definierende Vergleichsausdrücke) der Interpretation zugeordnet.

Für einen definierenden Bereichsausdruck wird bei der Auswertung überprüft, ob die Belegung mit einem Tupel aus der entsprechenden Relation des Zustands (syntaktisch) übereinstimmt.

In definierenden Vergleichsausdrücken wird ebenfalls eine syntaktische Überprüfung der Werte auf Gleichheit vorgenommen. Das heißt hier insbesondere, daß dem Ausdruck ω = ω der Wert *wahr* zugewiesen wird.

In gewöhnlichen Bereichs- und Vergleichsausdrücken erfolgt die Zuordnung eines Wertes der dreiwertigen Logik unter der Annahme, daß verschiedene Vorkommen von ω unterschiedliche Werte repräsentieren können. Damit wird abgesichert, daß keine unzulässigen Identifizierungen erfolgen, ohne daß Belegungen mit partiellen Tupeln ausgeschlossen werden müssen.

Bemerkung: Weitergehende Identifizierungsmöglichkeiten von ω-Vorkommen betrachten wir später. Die eingeschränkte Betrachtungsweise erfolgt hier im Hinblick auf Äquivalenzaussagen für den TRC und die Relationenalgebra.

Bei der Auswertung der einfachen TRC-Anfrage (x)/ R x $\wedge$ S x aus Beispiel 4.5 erhalten wir daher mit der Belegung (1,ω) von x zwei Interpretationen, für die sich *true* $\wedge$ *unknown* (R x definierend) bzw. *unknown* $\wedge$ *true* (S x definierend) bei der Auswertung ergibt. Die Auswertung liefert damit insgesamt eine leere gesicherte Antwort.

Die genaue Interpretationsvorschrift lautet wie folgt:

Interpretationsvorschrift

- Ein definierender Bereichsausdruck R x hat unter einer Interpretation I den Wert *wahr*, falls I(x) $\in$ I(R), und *falsch,* sonst.
- Ein gewöhnlicher Bereichsausdruck R(x) hat unter einer Interpretation I den Wert von I(x) $\in$ I(R), falls I(x) total ist. Ist I(x) nicht total, dann hat R(x) den Wert *unbekannt*, falls ($\exists$t $\in$ I(R))(I(x) Δ t), und *falsch*, sonst.

Bei Vergleichsausdrücken ist ebenfalls zu unterscheiden, ob es sich um einen gewöhnlichen Vergleichsausdruck handelt oder um einen definierenden Vergleichsausdruck, d.h. um einen Ausdruck, in dem einem Attribut einer vergleichsbeschränkten Variablen ein Wert zugewiesen wird.

- Definierende Vergleichsausdrücke können als Zuordnungen von Werten an die jeweilige beschränkte Variable angesehen werden. Stimmt der zugewiesene Wert mit dem Wert, der sich aus der Belegung der beschränkten Variablen ergibt, überein, hat der Ausdruck den Wert *wahr*. Dies gilt auch für den speziellen Wert ω.
- Für gewöhnliche Vergleichsausdrücke legen wir folgendes fest:
  Falls beide Operanden eines solchen Ausdrucks definiert sind, wird dem Ausdruck wie bei der gewöhnlichen Interpretationsvorschrift ein Boolescher Wert zugeordnet. Falls ein Operand oder beide Operanden undefiniert sind, erhält der Ausdruck den Wert *unbekannt*. Ausnahmen sind Vergleichsausdrücke, in denen ein Operand undefiniert und der andere der bezüglich der Vergleichsoperation minimale oder maximale Wert des entsprechenden Wertebereichs ist. In diesen Fällen wird den Ausdrücken in offensichtlicher Weise ein Boolescher Wert zugeordnet (wir beachten diese Sonderfälle im folgenden allerdings nicht weiter).
- Durch Induktion über den Aufbau von TRC-Ausdrücken dehnen wir die Definition auf beliebige TRC-Ausdrücke aus:

  I erfüllt $(\exists y)(\alpha) \Leftrightarrow_{df}$ es gibt eine Interpretation I', die sich von I höchstens in der Belegung von y und der Zuordnung von definierenden und gewöhnlichen Ausdrücken für y unterscheidet und die α erfüllt.

  I erfüllt $(\forall y)(\alpha) \Leftrightarrow_{df}$ jede Interpretation I', die sich von I höchstens in der Belegung von y und der Zuordnung von definierenden und gewöhnlichen Ausdrücken für y unterscheidet, erfüllt α.
- Für die Auswertung Boolescher Ausdrücke wird die dreiwertige Logik verwendet, die wir oben schon betrachtet haben. Eine Interpretation erfüllt einen Ausdruck, falls der zugehörige Boolesche Ausdruck gemäß der dreiwertigen Logik zu *wahr* ausgewertet wird.

Seien Interpretationen mit Festlegung von definierenden und gewöhnlichen Ausdrücken zur Unterscheidung von üblichen Interpretationen mit I_ω bezeichnet. Das durch die angegebene Interpretationsvorschrift festgelegte Verfahren zur Auswertung von TRC–Anfragen in ω-Zuständen wollen wir ω-Auswertung nennen.

Definition: Sei $q = (x_1,....,x_\ell)/\ \Phi(x_1,....,x_\ell)$ eine erlaubte TRC-Anfrage an ein DB-Schema σ, und sei z_ω ein ω-Zustand zu σ. Die ω-Antwort auf q in z_ω (in Zeichen: $\mathfrak{A}_\omega(q,z_\omega)$) ist wie folgt definiert:

falls $\ell \geq 1$: $\{(I_\omega(x_1),....,I_\omega(x_\ell)) \mid I_\omega$ Interpretation mit: I_ω erfüllt $\Phi\}$;

falls $\ell = 0$: *wahr* wenn für alle Interpretationen I_ω gilt: I_ω erfüllt Φ;
falsch wenn für keine Interpretation I_ω gilt: I_ω erfüllt Φ;

dabei sind die Interpretationen eindeutig bestimmt durch z_ω und q.

Beispiel 4.7: Seien R = {(1,2), (2,ω)} und S = {(2,3)} ω-Relationen über {A,B} und x eine Variable mit typ(x) = {A}.

a) Betrachte folgende erlaubte TRC-Anfrage:

$(x)/\ (\exists y)(R\ y \wedge x.A = y.B) \vee (\exists z)(S\ z \wedge x.A = z.B)$

Die ω-Auswertung liefert {(2), (ω), (3)} als Ergebnis. (ω) ist in der ω-Antwort enthalten, da bei Belegung von y mit (2,ω) der Bereichsausdruck R y den Wert *wahr* hat und der Vergleichsausdruck x.A = y.B als definierend für x angesehen wird.

b) Betrachte folgende erlaubte TRC-Anfrage:

$(x)/\ (\exists y)(R\ y \wedge x.A = y.B) \wedge \neg (\exists z)(S\ z \wedge x.A = z.B)$

Für diese Anfrage erhalten wir $\{(2)\}$ als ω-Antwort. Die Ersetzung der quantifizierten Teilausdrücke durch Disjunktionen ergibt für den Qualifikationsteil (andere als die für y und z angegebenen Belegungen müssen nicht betrachtet werden):

$(R(1,2) \wedge x.A = 2 \quad \vee \quad R(2,\omega) \wedge x.A = \omega) \quad \wedge \quad \neg (S(2,3) \wedge x.A = 3)$

$x.A = 2$ und $x.A = \omega$ sind jeweils als definierende Vergleichsausdrücke anzusehen, da $x.A = y.B$ in der Anfrage immer definierend sein muß (folgt aus der syntaktischen Struktur der Anfrage). Bei Belegung von x mit (2) erhalten wir mit der ω-Auswertung den Wert *wahr* für den Qualifikationsteil der Anfrage; bei Belegung mit (ω) wird zwar das erste Konjunktionsglied zu *wahr* ausgewertet, für das zweite Glied erhalten wir aber *unbekannt* als Wert, weshalb dem gesamten Ausdruck der Wert *unbekannt* zugewiesen wird. Für alle anderen Belegungen von x ergibt sich für das erste Konjunktionsglied und damit für den gesamten Ausdruck *unbekannt* als Wert.

Durch die Interpretationsvorschrift wird gewährleistet, daß die zu Beginn des Abschnitts angegebene Einschränkung der Belegungsmöglichkeiten für Variable in erlaubten Anfragen bei totalen DB-Zuständen auch hier vorgenommen werden kann: Jede Belegung einer Variablen y mit einem Tupel, das nicht aus den entsprechenden Relationen des Zustands stammt oder durch Vergleichsausdrücke in der Anfrage bestimmt ist, führt zu *falsch* als Wert in den zugehörigen definierenden Ausdrücken. Daher verhalten sich alle Belegungen mit Tupeln, die nicht aus der eingeschränkten Belegungsmenge stammen, "neutral", d.h. ihre Nicht-Beachtung hat keine Änderung des Ergebnisses zur Folge.

Die ω-Auswertung führt zu Antworten, in denen nur gesicherte Antworttupel enthalten sind. Die Vorkommen von ω in Antworttupeln haben nicht immer die gleiche Bedeutung wie die Vorkommen von ω in einem ω-Zustand. Der Grund hierfür ist die Definition von ω-Antworten als Mengen. Für einen ω-Zustand mit der Relation $R = \{(1,\omega), (2,\omega)\}$ erhalten wir etwa mit $q = (x)/\ (\exists y)(R\ y \wedge x.A = y.B)$ als ω-Antwort $\{(\omega)\}$. Zur Interpretation von Antworten muß daher die enge Ausdehnung statt der Vervollständigung als POSS-Funktion gewählt werden. Kein Unterschied besteht für die Vorkommen von ω in Zuständen und ω-Antworten bezüglich der Menge der möglichen Werte, die sie repräsentieren: Es sind stets alle Werte des entsprechenden Wertebereichs möglich.

Zur Vereinfachung betrachten wir im folgenden Anfragen mit genau einer Variablen in der Zielliste. Die Verallgemeinerung der Aussagen auf beliebige Anfragen stellt kein Problem dar.

Satz 4.1: Sei $q = (x)/\ \Phi(x)$ eine erlaubte TRC-Anfrage an ein DB-Schema σ. Dann gilt für jeden ω-Zustand z_ω zu σ:

$$\mathfrak{A}_\omega(q,z_\omega) \subseteq \mathcal{GA}_{uni}(q,z_\omega).$$

Beweis: Sei z_ω ein beliebiger ω-Zustand zu σ, und sei $t \in \mathfrak{A}_\omega(q,z_\omega)$. Es ist zu zeigen: Für jeden möglichen Zustand $\tilde{z}$ zu z_ω gibt es ein $\tilde{t} \in q(\tilde{z})$ mit $\tilde{t} \trianglerighteq t$, und zu jedem $\hat{t} \trianglerighteq t$ gibt es einen möglichen Zustand $\hat{z}$ zu z_ω mit $\hat{t} \in q(\hat{z})$. Dabei sind $q(\tilde{z})$ und $q(\hat{z})$ gemäß der gewöhnlichen Interpretationsvorschrift für TRC-Ausdrücke auszuwerten.

Betrachte den Ausdruck $\Phi(t)$. Da $\Phi(x)$ ein erlaubter TRC-Ausdruck ist, gehören für jede Interpretation I_ω zu jeder Variablen y in $\Phi(x)$ eindeutig bestimmte definierende Bereichsausdrücke und/oder Mengen definierender Vergleichsausdrücke. Für jede Interpretation I_ω kann ein zu $\Phi(t)$ unter der ω-Auswertung äquivalenter Boolescher Ausdruck $B_{\Phi(t)}$ über Vergleichsausdrücken und Konstanten wie folgt erhalten werden: Ersetze alle quantifizierten Teilausdrücke der Form $(\exists y)(\varphi(y))$ und $(\forall y)(\varphi(y))$ sukzessiv durch endliche Disjunktionen bzw. Konjunktionen mit Gliedern $\varphi(s)$, wobei s alle durch z_ω und φ eindeutig gegebenen Belegungen von y durchläuft und alle Zuordnungsmöglichkeiten von "definierend" und "gewöhnlich" für die Ausdrücke in φ berücksichtigt werden (wie in der Interpretationsvorschrift für quantifizierte Teilausdrücke gefordert).

Seien $\tilde{z}$ ein beliebiger möglicher Zustand zu z_ω und t" ein Tupel, das in $\tilde{z}$ als Belegung von x möglich ist und für das $t'' \trianglerighteq t$ gilt. Seien ferner I_ω eine Interpretation mit $I_\omega(x) = t$ und $\tilde{I}$ eine (gewöhnliche) Interpretation mit $\tilde{I}(x) = t''$.

Betrachte den Booleschen Ausdruck $\tilde{B}_{\Phi(t'')}$, der aus $\Phi(t'')$ analog wie $B_{\Phi(t)}$ aus $\Phi(t)$ gebildet werden kann: Ersetze alle quantifizierten Teilausdrücke der Form $(\exists y)(\varphi(y))$ und $(\forall y)(\varphi(y))$ sukzessiv durch endliche Disjunktionen bzw. Konjunktionen mit Gliedern $\varphi(s)$, wobei s alle durch $\tilde{z}$ und φ eindeutig bestimmten Belegungen von y durchläuft. Da $\tilde{z}$ ein totaler DB-Zustand ist, müssen definierende und gewöhnliche Vorkommen von Bereichs- und Vergleichsausdrücken nicht unterschieden werden. $\tilde{B}_{\Phi(t'')}$ und $B_{\Phi(t)}$ können sich nur dadurch unterscheiden, daß in Bereichs- und Vergleichsausdrücken von $\tilde{B}_{\Phi(t'')}$ definierte Werte vorkommen, wo in $B_{\Phi(t)}$ undefinierte Attributwerte stehen, und daß in $B_{\Phi(t)}$ in den erzeugten Disjunktionen und Konjunktionen weitere Glieder vorkommen. Diese zusätzlichen Glieder in $B_{\Phi(t)}$ können zum einen durch Alternativen bei der Zuordnung von definierenden und gewöhnlichen Ausdrücken bedingt sein. Zum anderen können für einzelne Belegungen von y Glieder hinzukommen, weil mehrere Tupel in einer Relation von z_ω durch die POSS-Funktion zu $\tilde{z}$ auf ein einziges Tupel der entsprechenden Relation in $\tilde{z}$ abgebildet werden.

Damit kann jede derartige in $B_{\Phi(t)}$ vorkommende Disjunktion oder Konjunktion eindeutig einer Disjunktion bzw. Konjunktion in $\tilde{B}_{\Phi(t'')}$ zugeordnet werden. Das gleiche gilt für die Glieder einer solchen Disjunktion oder Konjunktion. Insgesamt gehört daher zu jedem Bereichs- oder Vergleichsausdruck E von $B_{\Phi(t)}$ genau ein entsprechender Ausdruck $\tilde{E}$ in $\tilde{B}_{\Phi(t'')}$. Für die Belegungen einer Variablen y aus dem Ausdruck, auf den E und $\tilde{E}$ in $\Phi(x)$ zurückgehen, gilt: Die Belegung von y in $\tilde{E}$ ist eine Vervollständigung der Belegung von y in E.

Betrachten wir zur Erläuterung den Ausdruck $(\exists y)(R\ y \wedge S\ y)$ als Beispiel. Seien

$$I_\omega(R) = \{(1,\omega)\}, \quad I_\omega(S) = \{(\omega,1)\}$$

und $\tilde{z}$ gegeben durch

$$\tilde{z}(R) = \{(1,1)\} \text{ und } \tilde{z}(S) = \{(2,1)\}.$$

Wir erhalten (die definierenden Ausdrücke sind entsprechend gekennzeichnet):

$$R(1,\omega)_d \wedge S(1,\omega) \vee R(1,\omega) \wedge S(1,\omega)_d \vee R(\omega,1)_d \wedge S(\omega,1) \vee R(\omega,1) \wedge S(\omega,1)_d$$

und

$$R(1,1) \wedge S(1,1) \vee R(2,1) \wedge S(2,1).$$

Die Zuordnung der Konjunktionsglieder der beiden Disjunktionen ergibt sich aus den Belegungen.

Wir zeigen im folgenden zunächst, daß ein Ausdruck E

- stets den gleichen Wert wie der zugeordnete Ausdruck $\tilde{E}$ hat, wenn der Wert von E ungleich *unbekannt* ist

oder

- zu einer Belegung gehört, die für eine Variable nicht betrachtet werden muß, d.h. die keinen Einfluß auf das Ergebnis hat.

Betrachten wir als erstes Bereichsausdrücke in $B_{\Phi(t)}$, die unter I_ω definierend sind. Hat ein solcher Ausdruck E in $B_{\Phi(t)}$ den Wert *wahr*, dann hat gemäß den Möglichkeiten zur Belegung von Variablen auch der Ausdruck $\tilde{E}$ den Wert *wahr*: E, das aus einem R y in $\Phi(x)$ entstanden ist, kann gemäß der Interpretationsvorschrift nur *wahr* sein für I_ω, wenn $I_\omega(y) \in I_\omega(R)$ gilt; da das Vorkommen von y in $\tilde{E}$ mit einer Vervollständigung von $I_\omega(y)$ belegt ist, die in $\tilde{z}(R)$ vorkommt (dies ergibt sich aus der Zuordnung von E und $\tilde{E}$), hat auch E den Wert *wahr*.

Hat ein definierender Bereichsausdruck $E = R(I_\omega(y))$ in $B_{\Phi(t)}$ den Wert *falsch*, dann gibt es in $I_\omega(R)$ kein Tupel, das mit $I_\omega(y)$ syntaktisch übereinstimmt. Hat der entsprechende Ausdruck $\tilde{E} = R(t)$ in $\tilde{B}_{\Phi(t")}$ den Wert *wahr*, dann gibt es ein mit $I_\omega(y)$ verträgliches Tupel t' in $I_\omega(R)$, das von t überdeckt wird. Damit gibt es einen definierenden Bereichsausdruck E', der ebenfalls $\tilde{E}$ zugeordnet ist und den Wert *wahr* hat (es sei daran erinnert, daß jedes $t' \in I_\omega(R)$ genau ein $t \in \tilde{I}(R)$ "erzeugt"). Die Zuordnung von Ausdrücken in $B_{\Phi(t)}$ und $\tilde{B}_{\Phi(t")}$ bleibt demnach surjektiv. Die Belegung und Festlegung von "definierend", die zu E geführt hat, hat keine Bedeutung für den Wert von $B_{\Phi(t)}$ und kann unberücksichtigt bleiben.

Für gewöhnliche Bereichsausdrücke muß nur der Fall betrachtet werden, daß die Belegung der zugehörigen Variablen ein nicht totales Tupel ist. In diesem Fall wird der Wert *falsch* zugeordnet, falls es kein mit $I_\omega(y)$ verträgliches Tupel in $I_\omega(R)$ gibt. Dann kann es auch in $I(R_i)$ kein Tupel geben, das eine Vervollständigung von $I_\omega(y)$ ist. Daher hat der entsprechende Bereichsausdruck in $\tilde{B}_{\Phi(t")}$ ebenfalls den Wert *falsch*.

Betrachten wir nun definierende Vergleichsausdrücke. Solche Ausdrücke können nur den Wert *wahr* oder *falsch* haben. Sei y.B = w.C ein solcher Ausdruck in $\Phi(x)$ mit: Unter I_ω wird der B-Wert von y definiert. Falls $I_\omega(y)(B)$ und $I_\omega(w)(C)$ beide definiert und gleich sind, erfüllt I_ω den Ausdruck y.B = w.C. Da $\tilde{I}(y)$ eine Vervollständigung von $I_\omega(y)$ und $\tilde{I}(w)$ eine Vervollständigung von $I_\omega(w)$ ist, hat der entsprechende Ausdruck in $\tilde{B}_{\Phi(t")}$ ebenfalls den Wert *wahr*.

y.B = w.C hat unter I_ω auch den Wert *wahr*, wenn $I_\omega(y)(B) = I_\omega(w)(C) = \omega$. Die Interpretationen für totale DB-Zustände sind so definiert, daß Variable, die über Vergleichsausdrücke positiv beschränkt sind, nicht direkt belegt werden müssen, sondern ihre Belegung von den anderen Operanden in den Vergleichsausdrücken "übernehmen". Solche Vergleichsausdrücke haben daher trivialerweise den Wert *wahr*. In diesem Fall ist ein $\tilde{I}$ mit $\tilde{I}(x) = t"$ zu wählen, für das $\tilde{I}(y)(B) = \tilde{I}(w)(C)$ gilt. Eine solche Wahl ist immer möglich, da der Attributwert von y für die Interpretation I_ω nur einmal, und zwar in dem betrachteten Vergleichsausdruck, "definiert" wird.

Hat ein gewöhnlicher Vergleichsausdruck in $B_{\Phi(t)}$ unter I_ω einen definierten Wert, dann sind beide Operanden definiert und der entsprechende Ausdruck in $\tilde{B}_{\Phi(t'')}$ ist identisch mit dem Ausdruck für jede geeignete Interpretation $\tilde{I}$.

Aus diesen Betrachtungen folgt, daß es eine Interpretation $\tilde{I}$ zu $\tilde{z}$ gibt, so daß für jeden Ausdruck in $B_{\Phi(t)}$, der unter I_ω einen definierten Wert hat, der zugehörige Ausdruck in $\tilde{B}_{\Phi(t'')}$ den gleichen definierten Wert hat. $B_{\Phi(t)}$ und $\tilde{B}_{\Phi(t'')}$ unterscheiden sich damit nur dadurch, daß in $B_{\Phi(t)}$ Ausdrücke vorkommen, die den Wert *unbekannt* haben, und daß in $B_{\Phi(t)}$ zusätzliche Disjunktionen und/oder Konjunktionen eventuell mit zusätzlichen Gliedern vorkommen.

Nach Annahme erfüllt I_ω den Ausdruck $\Phi(t)$; damit erfüllt I_ω auch $B_{\Phi(t)}$. Wird in einem beliebigen Ausdruck der dreiwertigen Logik, der den Wert *wahr* hat, ein Vorkommen des Wertes ω durch *wahr* oder *falsch* ersetzt, dann ändert sich der Wert des Ausdrucks nicht, er bleibt *wahr*. Dies folgt unmittelbar aus den Definitionen der Junktoren (*Monotonieeigenschaft* der dreiwertigen Logik). Für alle Vergleichsausdrücke in $B_{\Phi(t)}$, denen gemäß ω-Auswertung der Wert *wahr* oder *falsch* zugeordnet wird, gilt: Dem entsprechenden Vergleichsausdruck in $\tilde{B}_{\Phi(t'')}$ wird für alle durch t'' und $\tilde{z}$ festgelegten Interpretationen der gleiche Wert zugeordnet. Damit erhalten wir, daß $\tilde{B}_{\Phi(t'')}$ und damit auch $\Phi(t'')$ den Wert *wahr* haben, wenn $B_{\Phi(t)}$ den Wert *wahr* hat.

$\tilde{z}$ wurde als beliebiger möglicher Zustand zu z_ω angenommen und t'' als beliebige Vervollständigung von t gewählt, die in $\tilde{z}$ als Belegung von x möglich ist. Zu jeder Vervollständigung von t gibt es offensichtlich einen möglichen Zustand, in dem diese Vervollständigung als Belegung von x genommen werden kann. Wir finden also für alle möglichen Zustände $\tilde{z}$ ein $t'' \in q(\tilde{z})$ mit $t'' \trianglerighteq t$, und jede Vervollständigung von t ist in einer möglichen Antwort auf q enthalten. □

Durch die ω-Auswertung wird einem Vergleich $\omega \ominus \omega$, der durch Einsetzung aus einem gewöhnlichen Vergleichsausdruck erhalten wird, immer der Wert *unbekannt* zugeordnet, d.h. auch dann, wenn beide Vorkommen von ω aus demselben Attribut eines Tupels des Zustands oder einer Belegung t einer Variablen stammen. Mit der Vervollständigung als POSS-Funktion kann geschlossen werden, daß ein solcher Vergleich unter diesen Voraussetzungen einen Booleschen Wert erhalten kann. Als Beispiele für beide Fälle betrachte die folgenden Anfragen an ein geeignetes DB-Schema:

$(x)/\ R\ x \wedge (\exists y)(R\ y \wedge x.A = y.A)$

$(x)/\ (\exists y)((R\ y \vee S\ y) \wedge (\exists z)(z.A = y.A \wedge x.A = z.A \wedge x.A = y.A))$

Für die eingeführte Interpretationsvorschrift sind die genannten Eigenschaften stets überprüfbar, so daß folgende Änderung der ω-Auswertung erfolgen kann:

Erweiterte Form der ω-Auswertung

Einem unter einer Interpretation I_ω gewöhnlichen Vergleichsausdruck $x.A \ominus y.B$ mit: x und y werden durch I_ω in A bzw. B mit ω belegt, wird für $\ominus \in \{\leq, =, \geq\}$ der Wert *wahr* und für $\ominus \in \{<, \neq, >\}$ der Wert *falsch* zugeordnet, falls folgendes gilt: Beide Vorkommen von ω stammen aus dem gleichen Attribut eines Tupels einer Relation des Zustands, oder sie stammen aus dem gleichen Attribut einer durch I_ω gegebenen Belegung einer Variablen.
In allen anderen Fällen bleibt die Interpretationsvorschrift unverändert.

Beispiel 4.8: Sei R = {(1,3), (2,ω)} eine Relation über {A,B}. Betrachte die folgende erlaubte Anfrage:

(x)/ R x ∧ ¬(∃y)(R y ∧ y.A ≥ x.A ∧ y.B < x.B)

Die beiden Vergleichsausdrücke sind gewöhnliche Vergleichsausdrücke (schon wegen der Operatoren). Wird x mit (1,3) belegt, erhalten wir für den quantifizierten Teilausdruck:

(*wahr* ∧ 1 ≥ 1 ∧ 3 < 3 ∨ *wahr* ∧ 2 ≥ 1 ∧ ω < 3) ≡
falsch ∨ *unbekannt* ≡
unbekannt

Für die Belegung von x mit (2,ω) ergibt sich:

(*wahr* ∧ 1 ≥ 2 ∧ ω < 3 ∨ *wahr* ∧ 2 ≥ 2 ∧ ω < ω) ≡
falsch ∨ *falsch* ≡
falsch

Unter Berücksichtigung der Negation ergibt sich damit {(2,ω)} als Antwort.

4.3.2 sub-Auswertungen

Wie zu Beginn von Abschnitt 4.3.1 begründet, geht im Fall totaler DB-Zustände für erlaubte Anfragen keine Information verloren, wenn nur Interpretationen betrachtet werden, in denen Variable mit Tupeln belegt werden, die entweder in den entsprechenden Relationen des Zustands vorkommen oder über Vergleichsausdrücke aus Konstanten der Anfrage oder indirekt aus Werten von Tupeln des Zustands gebildet werden können. Das heißt, zusätzliche Interpretationen, die durch Belegungen von Variablen mit beliebigen Tupeln aus dem beschränkten oder unbeschränkten Universum gegeben sind, haben keinen Einfluß auf das Ergebnis.

Im Fall partieller DB-Zustände gilt bei Anwendung der ω-Auswertung ebenfalls: ω-Tupel, die aus dem beschränkten Universum als Belegungen hinzukommen, können wegen der positiven Beschränktheit aller Variablen keinen Einfluß auf das Ergebnis haben. Definierende Bereichsausdrücke werden für solche Belegungen zu *falsch* ausgewertet, so daß diese Belegungen für die Bestimmung des Ergebnisses nicht betrachtet werden müssen.

In bezug auf bereichsbeschränkte, freie Variable einer Anfrage stellt eine solche Beschränkung von Belegungen keine echte Einschränkung für die Berechnung gesicherter uni-Antworten dar. Dies ergibt sich durch Verallgemeinerung von Lemma 4.1 und der folgenden, in Kapitel 2 erwähnten Eigenschaft gesicherter uni-Antworten: Sind zwei Tupel t", t' mit t" ≥ t' in der gesicherten uni-Antwort enthalten, dann sind auch alle Tupel t mit t" ≥ t ≥ t' in ihr enthalten.

Lemma 4.1: Sei q = (x)/ Φ(x) eine erlaubte TRC-Anfrage an ein DB-Schema σ, in der die Variable x durch genau einen Bereichsausdruck R x positiv beschränkt ist. Dann gilt für alle ω-Zustände z_ω zu σ:

$t \in \mathcal{GA}_{uni}(q,z_\omega) \Rightarrow t \in z_\omega(R) \vee (\exists t'', t' \in z_\omega(R))(t'' \neq t' \wedge t'' \geq t \geq t')$.

Beweis: Sei t ein Tupel aus dem beschränkten Universum zu q und z_ω, das nicht in $z_\omega(R)$ enthalten ist. $\mathcal{GA}_{uni}(q,z_\omega)$ ist eine Teilmenge von $\mathcal{GT}(q,z_\omega)$. Um in $\mathcal{GT}(q,z_\omega)$ enthalten sein zu können, muß t von einem Tupel t" aus $z_\omega(R)$ überdeckt werden. Sei in $z_\omega(R)$ kein Tupel t' mit t ≥ t' enthalten. Dann gibt es eine Vervollständigung von t, die für kein Tupel aus $z_\omega(R)$ eine Vervollständigung darstellt

und damit in keiner möglichen Relation zu R vorkommt. Gemäß der Definition gesicherter uni-Antworten muß diese Vervollständigung von t aber in einer möglichen Antwort zu q enthalten sein, damit sich t für die gesicherte uni-Antwort qualifiziert. □

Um Formen von Antworten erhalten zu können, die keine Teilmengen gesicherter uni-Antworten darstellen, müssen demnach nicht nur Belegungen aus dem beschränkten Universum eine Rolle spielen, die nicht in Zustandsrelationen enthalten sind, sondern es ist auch die Interpretationsvorschrift hinsichtlich der Auswertung von Bereichsprädikaten und Vergleichsausdrücken geeignet abzuändern.

Beispiel 4.9: Seien R = {(1, 2), (1,3)} und S = {(1, ω)} ω-Relationen über der Attributmenge {A,B}. Für die TRC-Anfrage

$$(x)/\ R\ x\ \wedge\ \neg\ (\exists y)(S\ y\ \wedge\ y.B = x.B)$$

ist (1, ω) gesichertes Antworttupel. Um dieses Ergebnis durch Anwendung einer Interpretationsvorschrift erhalten zu können, muß die Belegung von x mit (1, ω) möglich sein. Außerdem muß mit der Belegung die Information verknüpft sein, daß es in jeder möglichen Relation zu R wenigstens zwei Tupel gibt, die Vervollständigungen dieser Belegung sind.

Die Erweiterung der Belegungsmöglichkeiten von Variablen kann natürlich auch für Vorkommen gebundener Variablen erfolgen, wodurch in manchen Fällen mit geeigneter Änderung der Interpretationsvorschrift eine bessere Annäherung an die gesicherte uni-Antwort erreicht werden kann, als dies mit der ω-Auswertung möglich ist.

Beispiel 4.10: Seien R und S wie in Beispiel 4.9. Betrachte folgende Anfrage:

$$(x)/\ S\ x\ \wedge\ (\exists y)(R\ y\ \wedge\ y.B \neq x.B)$$

Das Tupel (1,ω) qualifiziert sich für die gesicherte uni-Antwort, da in jedem möglichen Zustand der für ω eingesetzte Wert entweder von 2 oder von 3 verschieden ist. Um dieses Ergebnis erhalten zu können, bietet sich eine Erweiterung der Belegungsmöglichkeiten für die gebundene Variable y an.

Im folgenden entwickeln wir eine Interpretationsvorschrift mit erweiterten Belegungsmöglichkeiten für Variable der Form, die in den Beispielen 4.9 und 4.10 angesprochen wurde. Wir verwenden für die Interpretationen, für die diese Vorschrift anzuwenden ist, die Bezeichnung I_{sub}. Mit dieser Bezeichnung soll ausgedrückt werden, daß auch Tupel, für die Überdeckungen (engl.: subsumptions) in Relationen vorhanden sind, als Belegungen bei der Vorschrift eine Rolle spielen und nicht wie bei der ω-Auswertung unberücksichtigt bleiben.

Um die Betrachtungen zu vereinfachen, beschränken wir uns hier auf erlaubte Anfragen, in denen jede Variable durch genau einen atomaren Bereichsausdruck positiv beschränkt ist (mit anschließender Negation für universell quantifizierte Variable). Die bei der ω-Auswertung eingeführte Vorgehensweise für beliebige erlaubte Anfragen - Unterscheidung zwischen definierenden und gewöhnlichen Vorkommen von Bereichs- und Vergleichsausdrücken - läßt sich ohne Schwierigkeiten übertragen.

Für die Belegung von Variablen soll in Interpretationen die Möglichkeit der Differenzierung der Vorkommen von ω bestehen, um die oben angesprochene Information unterbringen zu können. In Beispiel 4.9 steht ω im Tupel (1, ω) als

Belegung von x (nicht als Tupel von S) für die beiden Werte 2 und 3, d.h. in jedem möglichen Zustand gibt es die beiden Tupel (1,2) und (1,3) als Überdeckungen von (1,ω). Um die Existenz solcher Überdeckungen zu vermerken, verwenden wir im folgenden die Mengenklammern als Index für entsprechende Vorkommen von ω in Belegungen von Variablen. Sei das beschränkte Universum derart erweitert, daß das Symbol $\omega_{\{\}}$ bei der Bildung von Tupeln wie ein Konstantensymbol aus der Anfrage verwendet wird. Sei ferner die Überdeckungsrelation für ω-Tupel auf Tupel mit Vorkommen von $\omega_{\{\}}$ erweitert, indem dieses Symbol wie ω gelesen wird. Wir lesen $\omega_{\{\}}$ auch dann wie ω, wenn wir DB-Tupel als Antworttupel betrachten (mit der für Vorkommen von ω in Antworten festgelegten Bedeutung).

Die Existenz zweier verschiedener überdeckender Tupel zu einem Tupel t ist nicht hinreichend für den Schluß, daß jedes Vorkommen von ω in t ein solches "mengenwertiges" Vorkommen ist. Betrachte etwa eine ω-Relation R = {(1,2,ω), (1,ω,3)}. Dann ist (1,ω,ω) ein von beiden Tupeln in R überdecktes Tupel, für das von keinem seiner beiden Vorkommen von ω angenommen werden darf, daß es in jedem möglichen Zustand zwei verschiedene Überdeckungen gibt (betrachte {(1,2,3)} als mögliche Relation zu R). Für ein gegebenes Belegungstupel t ist die oben angegebene Forderung aber für alle in ihm vorkommenden unbekannten Werte in linearer Zeit überprüfbar, wenn die Zeit in Abhängigkeit von der Anzahl der Tupel der betreffenden Relation gemessen wird: In einem einzigen Durchlauf kann für alle unbekannten Attributwerte in t die Existenz überdeckender Tupel mit den notwendigen Eigenschaften (unterschiedliche definierte Werte) überprüft werden.

Um die neuen Belegungsmöglichkeiten von Variablen zu berücksichtigen, erweitern wir zunächst die Vorschrift zur Auswertung von Bereichsprädikaten wie folgt:

Seien I_{sub} eine Interpretation und R x ein atomarer Bereichsausdruck.

$$
\begin{aligned}
I_{sub} \text{ erfüllt } R\ x \Leftrightarrow_{df}\ & I_{sub}(x) \in I_{sub}(R) \text{ oder} \\
& (\exists B \in typ(x))(I_{sub}(x)(B) = \omega_{\{\}} \wedge \\
& (\exists t', t'' \in I_{sub}(R))(t' \geq I_{sub}(x) \wedge t'' \geq I_{sub}(x) \wedge (\forall A \in typ(x)) \\
& ((I_{sub}(x)(A) = \omega_{\{\}} \Rightarrow (t'(A) \neq \omega \wedge t''(A) \neq \omega \wedge t'(A) \neq t''(A))) \\
& \wedge (I_{sub}(x)(A) = \omega \Rightarrow t'(A) = t''(A) = \omega))
\end{aligned}
$$

Beispiel 4.11: Sei eine Relation R = {(1,2,3),(2,2,4),(1,2,4)} gegeben. Interpretationen mit den folgenden Belegungen von x erfüllen dann R x:

$(1,2,\omega_{\{\}})$, $(\omega_{\{\}},2,4)$, $(\omega_{\{\}},2,\omega_{\{\}})$.

Dagegen kann R x zum Beispiel nicht mit der Belegung $(1,\omega_{\{\}},\omega_{\{\}})$ erfüllt werden.

Für erlaubte TRC-Anfragen müssen nur solche Tupel aus dem beschränkten Universum als Belegungen genommen werden, mit denen gemäß der Interpretationsvorschrift ein atomarer Bereichsausdruck erfüllt werden kann. Aus der obigen Vorschrift ergibt sich daher unmittelbar, daß nur Tupel als Belegungen betrachtet werden müssen, die durch Einsetzen von $\omega_{\{\}}$ für Attributwerte aus Tupeln in einer Relation des Zustands erhalten werden, in dem die Anfrage ausgewertet wird. Die Anzahl dieser Tupel ist für eine Relation exponentiell in der Anzahl der Attribute der Relation, da keine Einschränkungen bezüglich des Vorkommens von $\omega_{\{\}}$ in Attributen gemacht werden. In der Vorschrift wird aber zusätzlich gefordert, daß zu jeder solchen Belegungsmöglichkeit t zwei Tupel in der Relation vorhanden

sein müssen, die in allen Attributen übereinstimmen, in denen t einen von $\omega_{\{\}}$ verschiedenen Wert hat. Wenn n die Anzahl der Tupel in einer Relation ist, kann damit die Anzahl der zu betrachtenden Belegungen für diese Relation nicht größer als $\binom{n}{2}$ sein. Eine bessere obere Schranke können wir angeben, wenn wir die Interpretationsvorschrift vollständig vorgestellt haben.

Die Unterscheidung zwischen mengenwertigen ($\omega_{\{\}}$) und gewöhnlichen (ω) Vorkommen von ω in Belegungen kann bei der Auswertung von TRC-Ausdrücken ausgenutzt werden. Wir betrachten zunächst eine Form der Ausnutzung, bei der die verschiedenen Vorkommen von $\omega_{\{\}}$ nicht weiter unterschieden werden und insbesondere das Wissen über die definierten Werte in den überdeckenden Tupeln nicht genutzt wird. Verwendung findet nur das Wissen darüber, daß in jedem möglichen Zustand Alternativen für die Belegung einer Variablen mit unterschiedlichen Werten in dem betreffenden Attribut existieren.

Wir fügen den Wahrheitswerten der dreiwertigen Logik zwei weitere Werte hinzu: *lokal wahr* (LW) und *lokal falsch* (LF). *Lokal wahr* wird einem Vergleich zugeordnet, wenn gesichert ist, daß in allen möglichen Zuständen eine Interpretation existiert, die für die beteiligten mengenwertigen Vorkommen von ω zu einem Vergleich definierter Werte mit dem Wahrheitswert *wahr* führt. Entsprechend erhält ein Vergleich mit mengenwertigen Vorkommen von ω den Wert *lokal falsch*, wenn gesichert ist, daß in allen möglichen Zuständen eine Interpretation existiert, für die entsprechend ein Vergleich definierter Werte mit dem Wahrheitswert *falsch* erhalten wird.

Die beiden neuen Wahrheitswerte heißen *lokal*, weil Aussagen über Belegungsmöglichkeiten für mengenwertige ω-Vorkommen von weiteren solchen Vorkommen abhängig sein können. In derartigen Fällen muß ein zugeordneter Wert nachträglich zu *unbekannt* abgeändert werden.

Betrachten wir nacheinander die verschiedenen Formen von Vergleichen mit mindestens einem Wert $\omega_{\{\}}$ als Operand, die bei Einsetzung von Werten für Projektionsterme in Vergleichsausdrücken erhalten werden können. Sei c im folgenden stets eine beliebige Konstante aus einem passenden Wertebereich.

1. Fall: $\omega_{\{\}} \ominus c$, $c \ominus \omega_{\{\}}$, $\omega_{\{\}} \ominus \omega$, $\omega \ominus \omega_{\{\}}$, $\ominus \in \{<, \leq, >, \geq\}$;
da über die von $\omega_{\{\}}$ repräsentierten Werte nichts bekannt ist, muß jedem Vergleich einer dieser Formen der Wert *unbekannt* (U) zugeordnet werden. Ausnahmen ergeben sich, wenn die Konstante c minimales oder maximales Element des zugehörigen Wertebereichs ist. Dann kann wie bei der ω-Auswertung verfahren werden.

2. Fall: $\omega_{\{\}} = c$, $c = \omega_{\{\}}$, $\omega_{\{\}} = \omega$ oder $\omega = \omega_{\{\}}$;
da $\omega_{\{\}}$ mehr als einen Wert repräsentiert, gibt es in jedem möglichen Zustand mindestens eine Möglichkeit für die Belegung der entsprechenden Variablen, durch die der Vergleich mit c oder einer beliebigen für ω eingesetzten Konstanten zu *falsch* wird. Dem Vergleich wird daher der Wert *lokal falsch* (LF) zugeordnet.

3. Fall: $\omega_{\{\}} \neq c$, $c \neq \omega_{\{\}}$, $\omega_{\{\}} \neq \omega$ oder $\omega \neq \omega_{\{\}}$;
in diesem Fall gilt, daß in jedem möglichen Zustand mindestens eine Belegung für die zu dem Vorkommen von $\omega_{\{\}}$ gehörende Variable gewählt werden kann, so daß die Ungleichung den Wert *wahr* hat. Es kann somit der Wert *lokal wahr* (LW) zugeordnet werden.

4. Fall: $\omega_{\{\}} \Theta \omega_{\{\}}$, $\Theta \in \{<, \leq, =, \neq, \geq, >\}$;
wir haben angenommen, daß nichts über die beiden repräsentierten Mengen von Belegungen bekannt ist. Daher muß *unbekannt* als Wert zugeordnet werden.

Für alle übrigen Werte als Operanden soll die Zuordnung von *wahr*, *falsch* und *unbekannt* wie bei der ω-Auswertung erfolgen.

Für die Auswertung der prädikatenlogischen Ausdrücke wird eine fünfwertige Logik mit den Werten *wahr*, *falsch*, *unbekannt*, *lokal wahr* und *lokal falsch* benutzt. Für diese fünfwertige Logik sind die Junktoren ∨, ∧ und ¬ wie folgt definiert:

∨	W	F	LW	LF	U
W	W	W	W	W	W
F	W	F	LW	LF	U
LW	W	LW	LW	LW	LW
LF	W	LF	LW	U	U
U	W	U	LW	U	U

∧	W	F	LW	LF	U
W	W	F	LW	LF	U
F	F	F	F	F	F
LW	LW	F	U	LF	U
LF	LF	F	LF	LF	LF
U	U	F	U	LF	U

¬	
W	F
F	W
LW	LF
LF	LW
U	U

Die übrigen Junktoren sind über die Äquivalenzen der Booleschen Logik definiert. Wie sich leicht nachprüfen läßt, gelten das Assoziativ- und das Kommutativgesetz sowie die Regeln von De Morgan. Die Absorptionsgesetze gelten dagegen nicht. Wir erhalten das in Abbildung 4.1 angegebene doppelte Hasse-Diagramm für diese Logik (vergleiche solche Diagramme für andere Logiken in [Bel77] und [Fit91]). Es zeigt die beiden Ordnungen "Wissen" und "Wahrheit" für die fünf Werte der Logik.

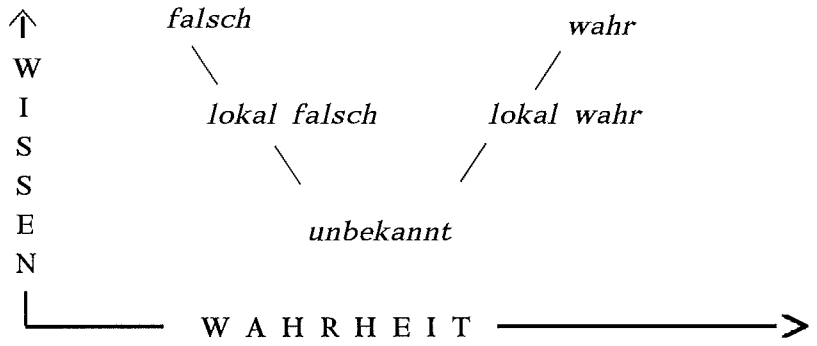


Abb. 4.1: Diagramm für die verwendete fünfwertige Logik

Wie bei der dreiwertigen Logik gilt auch bei dieser Logik eine Monotonieeigenschaft: Hat ein Ausdruck den Wert *wahr* oder *falsch*, dann kann in ihm jedes Vorkommen von *lokal wahr*, *lokal falsch* oder *unbekannt* in dem Ausdruck durch einen beliebigen anderen Wert ersetzt werden, ohne daß sich sein Wert ändert. Dies folgt unmittelbar aus den Definitionen der Junktoren.

Betrachten wir zunächst die Festlegung für ∨. Ohne die Werte *lokal wahr* und *lokal falsch* stimmt die Definition mit derjenigen der dreiwertigen Logik überein. Mit der "Wahrheits"-Ordnung gegeben durch *falsch*, *lokal falsch*, *unbekannt*, *lokal wahr*, *wahr* in dieser Reihenfolge, läßt sich ∨ auch wie folgt definieren:

$$a \vee b =_{df} \begin{cases} \textit{unbekannt} & \text{falls } a = b = \textit{lokal falsch} \\ \max(a,b) & \text{sonst} \end{cases}$$

Lokal falsch als Wert eines Ausdrucks bedeutet, daß der Wert *falsch* für diesen Ausdruck für bestimmte Belegungsmöglichkeiten von Variablen in jedem möglichen Zustand erhalten werden kann. Für zwei Teilausdrücke mit dem Wert *lokal falsch* können sich diese Belegungsmöglichkeiten für eine oder mehrere Variablen so unterscheiden, daß es einen möglichen Zustand gibt, in dem für alle in Frage kommenden Belegungen nicht beide Ausdrücke zugleich den Wert *falsch* haben. Demnach kann dem Ausdruck a $\vee$ b nicht der Wert *lokal falsch*, sondern nur *unbekannt* zugeordnet werden.

Für die übrigen Kombinationen von Werten besteht eine solche Abhängigkeit von den Belegungsmöglichkeiten nicht. Daher kann dort der Wert der Verknüpfung mit $\vee$ wie üblich als Maximum der Operanden gemäß der "Wahrheits"-Ordnung erfolgen.

Für die Festlegung von $\wedge$ gilt entsprechendes. Der Funktionstafel entspricht hier folgende Definition:

$$a \wedge b =_{df} \begin{cases} \textit{unbekannt} & \text{falls } a = b = \textit{lokal wahr} \\ \min(a,b) & \text{sonst} \end{cases}$$

Eine "Unverträglichkeit" von Belegungen kann dazu führen, daß in einem möglichen Zustand keine Belegung existiert, so daß zwei mit $\wedge$ verknüpfte Teilausdrücke beide den Wert *wahr* haben. Daher muß in diesem Fall der Wert *unbekannt* zugeordnet werden.

Bemerkung: Wegen der beiden Sonderfälle LF $\vee$ LF = U und LW $\wedge$ LW = U unterscheidet sich diese Logik schon für aussagenlogische Ausdrücke von allen bekannten mehrwertigen Logiken. Die Behandlung von Werten quantifizierter Teilausdrücke (siehe unten) stellt eine weitere Besonderheit dar.

Die Auswertung quantifizierter Teilausdrücke kann wegen der endlichen Anzahl möglicher Belegungen von Variablen wieder über endliche Disjunktions- bzw. Konjunktionsausdrücke erfolgen. Allerdings müssen die neu hinzukommenden Tupel mit Vorkommen von $\omega_{\{\}}$ gesondert betrachtet werden. Folgendes Beispiel veranschaulicht, worauf zu achten ist.

Beispiel 4.12: Sei eine Relation R = {(1,2), (1,3)} über {A,B} gegeben mit den ganzen Zahlen als Wertebereich für B. Betrachte die ja/nein-Anfrage

$$()/\ (\forall x)(R\ x \Rightarrow x.B \geq 0)$$

Neben den beiden Tupeln von R erhalten wir als mögliche Belegung von x das Tupel $(1,\omega_{\{\}})$. Da $\omega_{\{\}} \geq 0$ den Wert *unbekannt* erhält, ergibt sich für den gesamten Qualifikationsausdruck mit der fünfwertigen Logik ebenfalls der Wert *unbekannt*, wenn wir analog zur ω-Auswertung vorgehen.

Die Belegung von gebundenen Variablen mit den neu hinzukommenden Tupeln darf nicht zum Verlust gesicherter Information führen. Um dies zu gewährleisten, trennen wir in den für quantifizierte Teilausdrücke erzeugten Disjunktions- und Konjunktionsausdrücken die für die Tupel mit Vorkommen von $\omega_{\{\}}$ erzeugten Teilausdrücke von den übrigen Teilausdrücken ab. Das Ergebnis eines solchen abgetrennten Teilausdrucks wird mit dem Rest nur dann verknüpft, wenn diese Verknüpfung zu einem Wert führt, der gemäß der partiellen "Wissens"-Ordnung (*unbekannt* $\leq$ *lokal falsch* $\leq$ *falsch*, *unbekannt* $\leq$ *lokal wahr* $\leq$ *wahr*) größer ist als der Wert des Rests.

Hat ein Disjunktions- oder Konjunktionsausdruck, der für einen quantifizierten Teilausdruck erzeugt wurde, den Wert *lokal wahr* oder *lokal falsch*, dann wird der Wert für die weitere Auswertung in *wahr* bzw. *falsch* geändert, falls gilt: Der Wert ist nicht abhängig von einem $\omega_{\{\}}$-Vorkommen in der Belegung einer im Ausdruck freien Variablen. Dies Ersetzung ist unter der genannten Bedingung möglich, da Belegungen quantifizierter Variablen unabhängig voneinander erfolgen können.

Sei das durch die angegebene Interpretationsvorschrift festgelegte Verfahren zur Auswertung von TRC-Anfragen mit sub-Auswertung bezeichnet.

Wir sagen, daß eine Interpretation I_{sub} einen erlaubten TRC-Ausdruck Φ erfüllt, falls die sub-Auswertung den Wert *wahr* oder den Wert *lokal wahr* liefert.

Definition: Sei $q = (x_1,....,x_\ell)/\ \Phi(x_1,....,x_\ell)$ eine erlaubte TRC-Anfrage an ein DB-Schema σ, und sei z_ω ein ω-Zustand zu σ. Die sub-Antwort auf q (in Zeichen: $\mathfrak{A}_{sub}(q,z_\omega)$) ist wie folgt definiert:

falls $\ell \geq 1$: $\{(I_{sub}(x_1),....,I_{sub}(x_\ell)) \mid I_{sub}$ Interpretation mit: I_{sub} erfüllt $\Phi\}$;

falls $\ell = 0$: *wahr* wenn für alle Interpretationen I_{sub} gilt: I_{sub} erfüllt Φ;
falsch wenn für keine Interpretation I_{sub} gilt: I_{sub} erfüllt Φ;

dabei sind die Interpretationen eindeutig gegeben durch z_ω und q.

Beispiel 4.13: Bei der Auswertung der Anfrage aus Beispiel 4.9 (S. 73) erhalten wir mit $(1,\omega)$ als Belegung von x:

$(1,\omega_{\{\}}) \in \{(1,2),\ (1,3)\} \wedge \neg(\omega = \omega_{\{\}}) \equiv$

$W \wedge \neg LF \equiv$

$W \wedge LW \equiv$

LW

Die sub-Antwort ist demnach $\{(1,\omega)\}$.

Betrachten wir noch ein entsprechendes Beispiel "mit Semantik".

Beispiel 4.14: Seien die beiden folgenden Relationen zu einem entsprechenden DB-Schema gegeben:

EINKAUF

LNR	ABTNAME	ARTNR	EPREIS
L1	Hobby	A37	50.00
L1	Garten	A16	37.50
L2	Hobby	A37	49.80

SONDERANGEBOT

ARTNR	VPREIS
A37	ω
A16	37.50

Die Semantik erschließt sich mit der folgenden Anfrage:

"Gib die Namen aller Abteilungen, die Artikel einkaufen, die nicht mit dem Einkaufspreis als Verkaufspreis im Sonderangebot zu finden sind."

Eine mögliche Formulierung der Anfrage im TRC ist:

(x.ABTNAME)/ EINKAUF x $\wedge$ $\neg$ ($\exists$y: SONDERANGEBOT y)(y.ARTNR = x.ARTNR $\wedge$ y.VPREIS = x.EPREIS)

Mit der ω-Auswertung erhalten wir eine leere Antwort, mit der sub-Auswertung dagegen {(Hobby)}. Als zusätzliche Belegung von x wird unter anderem $(\omega_{\{\}}$,Hobby,A37,$\omega_{\{\}})$

betrachtet. Für diese Belegung erhalten wir für den quantifizierten Teilausdruck

$\neg$ (W $\wedge$ LF $\vee$ F $\wedge$ LF) $\equiv$
$\neg$ (LF) $\equiv$
LW

Satz 4.2: Sei q = (x)/ Φ(x) eine erlaubte TRC-Anfrage an ein DB-Schema σ; jede Variable in q sei durch genau einen Bereichsausdruck beschränkt. Sei z_ω ein beliebiger ω-Zustand zu σ. Dann gilt

$$\mathfrak{A}_\omega(q,z_\omega) \subseteq \mathfrak{A}_{sub}(q,z_\omega) \subseteq \mathcal{GT}(q,z_\omega).$$

Beweis: Die Gültigkeit von $\mathfrak{A}_\omega(q,z_\omega) \subseteq \mathfrak{A}_{sub}(q,z_\omega)$ folgt unmittelbar daraus, daß jede Interpretation I_ω auch als Interpretation I_{sub} angesehen werden kann und bei der Auswertung die hinzukommenden Belegungen nur dann Berücksichtigung finden, wenn ein Vorkommen von *unbekannt* durch einen gemäß der Wissensordnung größeren Wert ersetzt werden kann. Dabei geht die Monotonieeigenschaft der fünfwertigen Logik ein.

Die Vorschrift zur Auswertung atomarer Bereichsausdrücke gewährleistet, daß zu jeder Belegung einer Variablen mit Vorkommen von $\omega_{\{\}}$ in einem Attribut A mindestens zwei die Belegung überdeckende Tupel in der entsprechenden Relation des ω-Zustands gehören, die in A verschiedene definierte Werte haben. Jede Interpretation mit Vorkommen von $\omega_{\{\}}$ in einer Belegung kann daher als Repräsentant einer Menge von Interpretationen ohne solche Vorkommen angesehen werden.

Wegen Satz 4.1 und der getrennten Behandlung von Belegungen gebundener Variablen mit Vorkommen von $\omega_{\{\}}$ genügt es zu zeigen, daß die Vorkommen von $\omega_{\{\}}$ in Belegungen durch die Interpretationsvorschrift korrekt behandelt werden. Die Behandlung unterscheidet sich nicht von der Behandlung eines ω-Vorkommens bis auf die Auswertung von atomaren Bereichsausdrücken und Vergleichsausdrücken mit dem (Un)Gleichheitsoperator. Bei der Auswertung von atomaren Bereichsausdrücken, von denen wir hier nur definierende Ausdrücke betrachten müssen, wird festgestellt, ob es zu einer Interpretation I_{sub} mindestens zwei Interpretationen mit den genannten Eigenschaften gibt. Hier können sich demnach zusätzliche Belegungen gegenüber der ω-Auswertung qualifizieren. Zu einer solchen Belegung gehören damit aber auch mindestens zwei Interpretationen, die den betreffenden Bereichsausdruck bei der ω-Auswertung erfüllen. Erfüllt eine dieser Interpretationen den Qualifikationsausdruck der Anfrage, dann ist auch $I_{sub}(x)$ in der Menge gesicherter Antworttupel enthalten.

Betrachten wir nun Vergleichsausdrücke mit dem Gleichheitsoperator. Einem Vergleich c = $\omega_{\{\}}$ oder ω = $\omega_{\{\}}$ wird der Wert *lokal falsch* zugeordnet aufgrund folgender Überlegung: An dieser Stelle kann in jedem möglichen Zustand eine der I_{sub} zugeordneten Interpretationen gewählt werden, und die durch sie gegebenen Belegungen der Variablen können geeignet vervollständigt werden, so daß der Vergleichsausdruck den Wert *falsch* hat. Tritt ein weiterer derartiger Vergleichsausdruck auf, ist sein Wert in jedem möglichen Zustand abhängig von der schon getroffenen Wahl einer Interpretation für den anderen Vergleichsausdruck. Daher kann im allgemeinen nicht mehr garantiert werden, daß seine Auswertung den Wert *falsch* hat, wenn schon für den anderen Ausdruck der Wert *falsch* angenommen wird. Durch die eingeführte fünfwertige Logik wird garantiert, daß Boolesche Ausdrücke den Wert erhalten, den sie bei genau einer gewählten Interpretation mit dem gewünschten Effekt für einen Vergleichsausdruck haben. Daher wird

lokal falsch ∨ *lokal falsch* der Wert *unbekannt* zugeordnet; analog hat *lokal wahr* ∧ *lokal wahr* ebenfalls den Wert *unbekannt*.

Für Vergleichsausdrücke der Form $c \neq \omega_{\{\}}$ und $\omega \neq \omega_{\{\}}$ ergeben sich mit den getroffenen Festlegungen entsprechende Schlußfolgerungen.

Durch die Wertzuordnung und die spezielle verwendete Logik wird somit für jeden möglichen Zustand die Auswertung mit einer Interpretation nachgebildet, auf deren Existenz mit I_{sub} geschlossen werden darf. Die Auswertung erfolgt ansonsten in der gleichen Weise wie bei der ω-Auswertung. Mit Satz 4.1 ist damit dieser Satz bewiesen. □

Bei der sub-Auswertung wird strukturelles Wissen über die Anfrage verschenkt: Die Zuordnung der verschiedenen mengenwertigen Vorkommen von ω zu Attributen in den Variablen aus der Anfrage wird nicht genutzt. Dadurch kommt es immer dann zu einem Informationsverlust, wenn Belegungsmöglichkeiten in Vergleichsausdrücken und allgemein in Teilausdrücken unabhängig voneinander sind. Im Fall einer solchen Unabhängigkeit in Teilausdrücken (Form: x.A Θ y.B mit x ≠ y) kann einem Vergleich $\omega_{\{\}} = \omega_{\{\}}$ der Wert LF und entsprechend einem Vergleich $\omega_{\{\}} \neq \omega_{\{\}}$ der Wert LW zugewiesen werden. Im Fall einer solchen Unabhängigkeit für Teilausdrücke kann ein resultierender Ausdruck LW ∧ LW bzw. LF ∨ LF den Wert LW bzw. LF erhalten. Um diese Unabhängigkeit von Belegungen in Teilausdrücken ausnutzen zu können, müssen die Vorkommen der Werte LW und LF bei der Auswertung entsprechend gekennzeichnet werden.

Jedem Wert LW und LF wird dazu eine nichtleere Menge von Indizes hinzugefügt. Jeder Index ist eine nicht leere Teilmenge der Menge aller Variablen der Anfrage und repräsentiert eventuelle Belegungsabhängigkeiten der Variablen untereinander. Wird einem Vergleich der Wert LW oder LF nach obiger Festlegung zugeordnet, dann erhält dieser Wert zusätzlich eine Indexmenge, die als einziges Element eine ein- oder zweielementige Menge von Variablen enthält. Bei diesen Variablen handelt es sich um die Variable bzw. die Variablen des zugrundeliegenden Vergleichsausdrucks mit einem mengenwertigen Vorkommen von ω im entsprechenden Attribut. Bei Verknüpfungen dieser Werte ist das Ergebnis von den Indexmengen der Operanden abhängig. In der Indexmenge des Ergebnisses werden die sich aus der Verknüpfung und den Indizes der Operanden ergebenden Abhängigkeiten aufgenommen.

Seien X und Y im folgenden beliebige derartige Indexmengen über einer gegebenen Menge von Variablen. Wegen der Kommutativität der Operationen ∨ und ∧ können die Funktionstabellen in vereinfachter Form angegeben werden; für die mit * gekennzeichneten Stellen sind die Verknüpfungsresultate im Anschluß gesondert angegeben.

∨	W	F	LW_Y	LF_Y	U
W	W	W	W	W	W
F	W	F	LW_Y	LF_Y	U
LW_X	W	LW_X	*	LW_X	LW_X
LF_X	W	LF_X	LW_Y	*	U
U	W	U	LW_Y	U	U

∧	W	F	LW_Y	LF_Y	U
W	W	F	LW	LF_Y	U
F	F	F	F	F	F
LW_X	LW_X	F	*	LF_Y	U
LF_X	LF_X	F	LF_X	*	LF_X
U	U	F	U	LF_Y	U

¬	
W	F
F	W
LW_X	LF_X
LF_X	LW_X
U	U

Seien $X = \{X_1,....,X_m\}$ und $Y = \{Y_1,....,Y_n\}$. Mit X^Y bezeichnen wir diejenige Indexmenge, die alle Mengen $X_i \cup Y_j$, $i \in \{1,....,m\}$, $j \in \{1,....,n\}$, enthält, für die

$X_i \cap Y_j = \emptyset$ gilt. Dann sind die fehlenden Einträge in den Funktionstabellen wie folgt festgelegt:

$$LW_X \vee LW_Y = LW_{X \cup Y}$$

$$LF_X \vee LF_Y = \begin{cases} LF_{X \wedge Y} & \text{falls } X \wedge Y \neq \emptyset \\ U & \text{sonst} \end{cases}$$

$$LW_X \wedge LW_Y = \begin{cases} LW_{X \wedge Y} & \text{falls } X \wedge Y \neq \emptyset \\ U & \text{sonst} \end{cases}$$

$$LF_X \wedge LF_Y = LF_{X \cup Y}$$

Zur Begründung für diese Festlegung: Im ersten und im letzten Fall sind die Belegungsabhängigkeiten der Operanden unabhängig voneinander, da es für das Ergebnis ohne Bedeutung ist, welcher der Operanden den Wert *wahr* bzw. *falsch* annimmt. Im Ergebnis werden daher die Belegungsabhängigkeiten beider Operanden in die Indexmenge aufgenommen. In den beiden anderen Fällen muß für beide Operanden für eine Belegung der gleiche Wert erhalten werden. Da eine Variable für jeden Operanden eventuell anders belegt werden muß, um *wahr* bzw. *falsch* als Ergebnis zu erhalten, muß das Ergebnis der Verknüpfung immer dann als *unbekannt* angenommen werden, wenn die jeweiligen Belegungsabhängigkeiten der Operanden nicht leeren Schnitt haben. Da jede Indexmenge Alternativen solcher Abhängigkeiten darstellt, muß jede mögliche Kombination überprüft werden. Gibt es keine Kombination mit nicht leerem Schnitt, ist *unbekannt* als Wert herzunehmen; ansonsten bilden die Vereinigungen der Variablenmengen mit leerem Schnitt die Indexmenge des Ergebnisses.

Beispiel 4.15: Sei zusätzlich zu den Relationen R = {(1,2), (1,3)} und S = {(1,ω)} aus den vorangehenden Beispielen noch die Relation T = {(1,ω)} gegeben. Betrachte folgende Anfrage:

$$(x,y)/\ R\ x \wedge R\ y \wedge \neg(\exists v)(S\ v \wedge x.B = v.B \wedge y.B = v.B) \wedge \neg(\exists w)(T\ w \wedge x.B = w.B)$$

Mit der einfachen sub-Auswertung erhalten wir mit der Belegung $(1,\omega_{\{\}})$ für x und y:

$$W \wedge W \wedge \neg(LF \wedge LF) \wedge \neg(LF) \equiv$$
$$LW \wedge LW \equiv$$
$$U$$

Mit der indizierten sub-Auswertung ergibt sich (in vereinfachter Schreibweise):

$$W \wedge W \wedge \neg(LF_{\{x\}} \wedge LF_{\{y\}}) \wedge \neg(LF_{\{x\}}) \equiv$$
$$\neg(LF_{\{x\},\{y\}}) \wedge LW_{\{x\}} \equiv$$
$$LW_{\{x\},\{y\}} \wedge LW_{\{x\}} \equiv$$
$$LW_{\{y,x\}}$$

Wird als Ergebnis der Auswertung *lokal wahr* erhalten, dann wird dieses Ergebnis mit *wahr* gleichgesetzt unabhängig von der zugehörigen Indexmenge.

Betrachten wir ein weiteres Beispiel:

Beispiel 4.16: Seien die Relationen R = {(1,2), (1,3)}, S = {(1,4), (1,5)} und T = {(1,ω)} gegeben. Betrachte folgende Anfrage:

$$(x)/\ R\ x\ \wedge\ \neg(\exists y)(\exists z)(S\ y\ \wedge\ T\ z\ \wedge\ x.B \neq y.B\ \wedge\ x.B = z.B)$$

Für die Belegungen von x mit (1,2) und (1,3) erhalten wir wegen des Vergleichsausdrucks x.B = z.B jeweils den Wert *unbekannt* für den quantifizierten Teilausdruck. Mit der Belegung $(1,\omega_{\{\}})$ ergibt sich unter Verwendung der indizierten sub-Auswertung mit der gleichen Belegung für y:

$$\neg(LW_{\{x,y\}} \wedge LF_{\{x\}}) \equiv$$
$$\neg(LF_{\{x\}}) \equiv$$
$$LW_{\{x\}}$$

Damit qualifiziert sich $(1,\omega_{\{\}})$ für die Antwort, d.h. wir erhalten (1,ω) als Antworttupel.

Eine naheliegende Idee zur Verbesserung der beschriebenen Auswertungsverfahren ist die Berücksichtigung der definierten Werte aus den Überdeckungen: Statt keine Information oder nur Variable mit den mengenwertigen Vorkommen von ω zu verbinden, werden die konkreten definierten Werte in den zugehörigen Attributen der vorhandenen Überdeckungen mit diesen Vorkommen verknüpft. Damit können Vergleiche mit definierten Werten in einigen Fällen zu Booleschen Werten ausgewertet werden, in denen sie mit der indizierten oder einfachen sub-Auswertung den Wert *lokal wahr* oder *lokal falsch* erhalten.

4.3.3 Komplexität der Auswertungen

Bisher haben wir außer einigen Bemerkungen im Zusammenhang mit Belegungen noch nichts zur Komplexität der verschiedenen Auswertungsverfahren gesagt. Als Komplexitätsmaß für Anfragen wird gewöhnlich die *Datenkomplexität* benutzt ([Vas86]). Die Komplexität ist hierbei als eine Funktion der Größe von Datenbankzuständen definiert, während die Länge der Eingabe, d.h. also in unserem Fall die Länge der TRC-Ausdrücke, als fest angenommen wird. Für totale Zustände sind Antworten auf erlaubte TRC-Anfragen in polynomieller Zeit ermittelbar (was auch für Algebraanfragen - ohne Komplementbildung - gilt). Dies läßt sich einfach einsehen, wenn man die Auswertung von TRC-Anfragen gemäß der Interpretationsvorschrift nachvollzieht: Die Anzahl der Belegungsmöglichkeiten für die freien und die gebundenen Variablen ist durch die Anzahl der Tupel in den Relationen des betrachteten Zustands beschränkt. Zu jeder Belegungsmöglichkeit für die freien Variablen einer Anfrage wird aus dem Qualifikationsteil der Anfrage ein variablenfreier aussagenlogischer Ausdruck erzeugt. Die Länge dieses Ausdrucks ist von der Anzahl der Tupel in dem Datenbankzustand abhängig, für den die Anfrage ausgewertet wird. Für aussagenlogische Ausdrücke ohne freie Variablen kann der Wahrheitswert in linearer Zeit ermittelt werden. Da die Elimination von Duplikaten im Ergebnis mit quadratischem Aufwand (bzw. besser, durch Einsatz eines geeigneten Sortieralgorithmus) erfolgen kann, ergibt sich insgesamt eine polynomielle Komplexität für die Auswertung.

Bei der ω-Auswertung ergibt sich komplexitätsmäßig kein Unterschied zur gewöhnlichen Auswertung: Die Belegungsmöglichkeiten für die Variablen sind wie im gewöhnlichen Fall festgelegt. Die Unterscheidung von Bereichs- und Ver-

gleichsausdrücken in definierende und gewöhnliche Ausdrücke bewirkt eine Multiplikation der Anzahl der Belegungen mit einer Größe, die durch die Anzahl der zu beachtenden Ausdrücke beschränkt ist. Die Zuordnung von Werten zu Vergleichsausdrücken geschieht analog zur gewöhnlichen Auswertung, und die Verwendung der dreiwertigen Logik statt der Booleschen Logik erfordert keine höheren Kosten.

Für die beiden Formen der I_{sub}-Auswertung wollen wir zunächst betrachten, welche Tupel als Belegungen berücksichtigt werden müssen. Oben haben wir schon eine obere Schranke für ihre Anzahl angegeben. Unter Berücksichtigung der Interpretationsvorschrift kann diese Schranke verbessert werden.

Sei $q = (x)/ R\ x \wedge \Phi(x)$ eine erlaubte TRC-Anfrage an ein DB-Schema σ. Betrachte ein beliebiges Tupel t der ω-Relation $z_\omega(R)$ des Zustands z_ω zu σ, in dem q ausgewertet werden soll. Sei T die Menge von Tupeln mit Vorkommen von $\omega_{\{\}}$, die für t als zusätzliche Belegungen wie oben beschrieben erzeugt werden, indem t mit allen anderen Tupeln von $z_\omega(R)$ verglichen wird. Qualifiziert sich ein Tupel $t_i \in T$ gemäß der sub-Auswertung für die Antwort, dann gibt es ein Attribut $A \in typ(x)$ mit $t_i(A) = \omega_{\{\}}$, so daß folgendes gilt:

Es qualifiziert sich auch jedes Tupel $t_j \in T$ mit $t_j(A) = \omega_{\{\}}$, das eine Überdeckung von t_i ist.

Betrachte zwei derartige Tupel t_i, t_j, wobei t_i in mindestens einem weiteren Attribut den Wert $\omega_{\{\}}$ hat. t_j hat einen definierten Wert in mindestens einem Attribut B verschieden von A, in dem t_i den Wert $\omega_{\{\}}$ hat. Werden bei der Auswertung des Ausdrucks $\Phi(t_i)$ zwei logische Werte miteinander verknüpft, die sich bei Einsetzung eines definierten Wertes im Attribut A oder B von t_i ändern, dann kann diese Änderung höchstens zu einer Änderung des für $\Phi(t_i)$ erhaltenen Wertes führen, wenn dieser *lokal wahr* ist. In diesem Fall kann ein Änderung zu *wahr* erfolgen. Dies ergibt sich zum einen aus der Monotonieeigenschaft der fünfwertigen Logik und zum anderen aus der Festlegung LF $\vee$ LF = U und LW $\wedge$ LW = U. Aufgrund der Monotonieeigenschaft kann die Zuordnung von *wahr* oder *falsch* statt *unbekannt* für einen Vergleich keine Änderung des Gesamtergebnisses bewirken, wenn dieses verschieden von *unbekannt* ist. Die genannte Festlegung sichert ab, daß ein Wert ungleich *unbekannt* für einen Ausdruck nur erhalten werden kann, wenn keine Abhängigkeit dieses Wertes von mehr als einem Vorkommen von $\omega_{\{\}}$ in einem Vergleich gegeben ist. Falls eine Abhängigkeit des Wertes von $\Phi(t_i)$ von einem Vorkommen von $\omega_{\{\}}$ in einem Attribut gegeben ist, ist dieses Attribut somit eindeutig bestimmt. Falls keine solche Abhängigkeit vorhanden ist, qualifiziert sich t für die Antwort und die Belegungen aus T müssen nicht betrachtet werden.

Damit muß jedes Attribut $A \in typ(x)$ (= Attr(R)) höchstens einmal in der für t erzeugten Tupelmenge durch ein Tupel t_i mit $t_i(A) = \omega_{\{\}}$ "vertreten" sein, und wir erhalten als obere Schranke für die Anzahl zusätzlich zu betrachtender Belegungen für die Relation R: $\min(\binom{n}{2}, |Attr(R)| * n)$; dabei sei n wieder die Anzahl der Tupel in $z_\omega(R)$.

Die Schranke gilt in gleicher Weise für die Belegungen quantifizierter Variablen, so daß wir insgesamt feststellen können:

Satz 4.3: Die Anzahl der Belegungen, die bei der sub-Auswertung für eine Variable einer Anfrage zu betrachten sind, ist beschränkt durch $\min(\binom{n}{2}, |Attr(R)| * n)$. Dabei ist R die Relation aus dem Bereichsausdruck, durch den die Variable beschränkt ist; n ist die Anzahl der Tupel von R in dem DB-Zustand, in dem die Anfrage ausgewertet wird.

Die Auswertung selbst ergibt in ihrer einfachen Variante, d.h. ohne Indizierung von Werten, gegenüber der gewöhnlichen Auswertung und der ω-Auswertung keine höheren Kosten. Die indizierte Variante kann einen Mengenabgleich erfordern. In diesen Mengen sind aber höchstens soviele Elemente enthalten, wie Variable in der Anfrage vorkommen, so daß diese Kosten vernachlässigt werden können.

4.4 Auswertung von Algebraausdrücken

In Abschnitt 4.1 haben wir schon gesehen, welche Probleme beim Auswerten von Ausdrücken der Relationenalgebra entstehen können, wenn bei der Definition der Operationen verschiedene Vorkommen von ω-Tupeln, die sich in ihren definierten Werten nicht unterscheiden, als Vorkommen identischer Tupel angesehen werden. Aus den in Abschnitt 4.2 angeführten Schlußfolgerungen von Ergebnissen aus [IL84] ergibt sich allgemein, daß keine Definition der Operationen angegeben werden kann, die gesicherte Information im Ergebnis für beliebige Ausdrücke unter Beibehaltung des Funktionalitätsprinzips für die Auswertung von Ausdrücken garantiert. Es muß also von ω-Relationen zu erweiterten Formen solcher Relationen übergegangen oder das Funktionalitätsprinzip aufgegeben werden, wenn gesicherte Information für beliebige Algebraausdrücke erhalten werden soll.

4.4.1 Die sm- und die switch-Methode

Eine erste Erweiterung, die wir betrachten wollen, ist die Unterscheidung der Tupel jeder Relation in *gesicherte* und *mögliche* Tupel. Auf der Basis dieser Unterscheidung von Tupeln in ω-Relationen wurden in [Bis83] unter Annahme der engen Ausdehnung als Z-POSS-Funktion erweiterte Definitionen für Operationen der Relationenalgebra vorgeschlagen. Wir orientieren uns an diesen Definitionen, ohne allerdings die in [Bis83] stets durchgeführte Überführung von Ergebnissen in nicht redundante Relationen vorzunehmen. Diese Reduktion ist für die Semantik der Operationen ohne Belang. Außerdem führen wir keine Bereichsdeklarationen in den Relationen mit, da sie nur für die Betrachtung von Eigenschaften benötigt werden, die hier keine Rolle spielen (*hinreichend* und *beschränkt*, siehe Kapitel 5, Abschnitt 5.1). Eine weitere Vereinfachung gegenüber [Bis83] besteht darin, daß wir für alle Mengenoperationen gleichen Typ für die Operanden verlangen.

Die in den Operationen vorgenommene Elimination von Duplikaten hat zur Folge, daß als POSS-Funktion für die Interpretation von Antworten wie im vorigen Abschnitt die enge Ausdehnung hergenommen werden muß.

Seien (RT, α) ein Relationstyp mit Wertebereichsfunktion dom und STATUS $\notin \alpha$ ein Attribut mit Wertebereich {s,m}, wobei s für *gesichert* und m für *möglich* steht. Wir gehen über zu einem erweiterten Relationstyp (RT, $\alpha \cup$ {STATUS}) mit entsprechend angepaßter Wertebereichsfunktion. Eine ω-Relation zu diesem erweiterten Typ ist eine ω-Relation R über $\alpha \cup$ {STATUS} mit folgender Eigenschaft: Für alle Tupel t aus R ist t(STATUS) definiert.

Seien im folgenden alle ω-Relationen stets derartig erweiterte ω-Relationen. Für die Relationen eines Zustands soll gelten, daß alle Tupel im Attribut STATUS den Wert s haben. Für die Attributmenge $\alpha \cup$ {STATUS} schreiben wir auch α+.

Operationen der Relationenalgebra, in denen nur die Gleichheit als Vergleichsrelation auf Wertebereichen eine Rolle spielt, können auf ω-Relationen über α bzw. α+ wie auf totale Relationen angewandt werden, wenn ω jedem Wertebereich

als neuer Wert hinzugefügt wird und die jeweilige Gleichheitsrelation um $\omega = \omega$ ergänzt wird. Wir verwenden einige der Operationen mit der sich durch diese Festlegung ergebenden Bedeutung zur Vereinfachung von Definitionen.

Die Operation "Umbenennung" wird in keiner der folgenden Definitionen erwähnt, da sie nur auf der Typebene wirkt.

Die erweiterten Operationen der Relationenalgebra sind wie folgt definiert:
Seien zunächst R und U ω-Relationen über der gleichen Attributmenge $\alpha+$.

1) Vereinigung: $R \cup_e U =_{df} R \cup U$

2) Differenz:
$$R \setminus_e U =_{df} \{t \mid t \in R \wedge \neg(\exists t' \in U)(t'_{|\alpha} \,\Delta\, t_{|\alpha})\} \cup$$
$$\{t \mid t(\text{STATUS}) = m \wedge (\exists t' \in R)(t'_{|\alpha} = t_{|\alpha}) \wedge (\exists u' \in U)(u'_{|\alpha} \,\Delta\, t_{|\alpha})$$
$$\wedge (\forall u'' \in U)(\text{Def}(u'') = \alpha \Rightarrow (u''_{|\alpha} \neq t_{|\alpha} \vee u''(\text{STATUS}) = m))\}$$

Beispiel: $\{(1,2,s), (1,3,m), (3,\omega,s), (2,\omega,m)\} \setminus_e \{(1,\omega,m), (1,3,s), (2,\omega,s)\} =$
$\{(1,2,m), (3,\omega,s), (2,\omega,m)\}$

3) Projektion: Sei $\beta \subseteq \alpha$; $R[\beta]_e =_{df} R[\beta \cup \text{STATUS}]$

4) Selektion: Sei vga ein einfacher Vergleichsausdruck der Form $A \ominus c$, $c \ominus A$ oder $A \ominus B$ mit $A,B \in \alpha$, $\text{dom}(A) = \text{dom}(B)$ und $c \in \text{dom}(A)$.

Ein ω-Tupel t über α erfüllt vga *mit Sicherheit*, falls t(A) definiert ist und in der angegebenen Vergleichsrelation zu c steht bzw. falls t(A) und t(B) definiert sind und in der angegebenen Vergleichsrelation stehen.

t erfüllt vga *möglicherweise*, falls t(A) bzw. t(A) und/oder t(B) gleich ω sind.

$$R[\text{vga}]_e =_{df} \{t \mid t \in R \wedge t \text{ erfüllt vga mit Sicherheit}\} \cup$$
$$\{t \mid t(\text{STATUS}) = m \wedge (\exists t' \in R)(t'_{|\alpha} = t_{|\alpha} \wedge$$
$$t' \text{ erfüllt vga möglicherweise})\}$$

Bemerkung: Randfälle wie $\max(\text{dom}(A)) \geq \omega$ seien hier nicht betrachtet.

Beispiel: Sei $R = \{(1,2,m), (1,3,s), (2,\omega,s)\}$ eine ω-Relation über {A,B,STATUS}; dann gilt $R[B=3] = \{(1,3,s), (2,\omega,m)\}$. Für $R[A=B]$ erhalten wir $\{(2,\omega,m)\}$.

Seien nun U und V ω-Relationen über $\beta+$ bzw. $\gamma+$ mit $\beta,\gamma \subseteq \alpha$.

5) Join:

$$U \bowtie_e V =_{df} \{t \mid t \text{ erweitertes } \omega\text{-Tupel über } \beta \cup \gamma \wedge (\exists t' \in U)(\exists t'' \in V)$$
$$(t_{|\beta\setminus\gamma} = t'_{|\beta\setminus\gamma} \wedge t_{|\gamma\setminus\beta} = t''_{|\gamma\setminus\beta} \wedge t_{|\beta\cap\gamma} = \sup(t'_{|\beta\cap\gamma}, t''_{|\beta\cap\gamma})$$
$$\wedge ((\beta \cap \gamma \subseteq \text{Def}(t') \cap \text{Def}(t'') \wedge t'_{|\beta\cap\gamma} = t''_{|\beta\cap\gamma} \wedge t'(\text{STATUS}) =$$
$$t''(\text{STATUS}) = t(\text{STATUS}) = s) \vee$$
$$(t'_{|\beta\cap\gamma} \,\Delta\, t''_{|\beta\cap\gamma} \wedge t(\text{STATUS}) = m \wedge (t'_{|\beta\cap\gamma} \neq t''_{|\beta\cap\gamma} \vee t'(\text{STATUS}) = m$$
$$\vee t''(\text{STATUS}) = m))))\}$$

Beispiel: Seien $U = \{(1,\omega,m), (2,2,s)\}$ und $V = \{(2,3,s), (\omega,4,s)\}$ ω-Relationen über {A,B}+ bzw. {B,C}+. Dann erhalten wir:

$$U \bowtie_e V = \{(1,2,3,m), (1,\omega,4,m), (2,2,3,s), (2,2,4,m)\}$$

Diese Definitionen weichen teilweise von den in [Bis83] gegebenen Definitionen ab. Zum einen ist die Selektion ausgedehnt worden auf beliebige einfache Vergleichsausdrücke und nicht beschränkt auf Ausdrücke der Form A = c und A = B. Zum anderen übernehmen wir bei der Definition der Selektion nicht die in einem Vergleich c = ω bzw. ω = c enthaltene Information, daß ein sich qualifizierendes Tupel im entsprechenden Attribut den Wert c haben muß. So erhält man mit der Definition der Selektion über den Join, wie sie in [Bis83]erfolgt, für R \ S[B=3] das Ergebnis R, wenn R = {(1,2,s)} und S = {(1,ω,s)}. Mit unserer Definition ergibt sich dagegen {(1,2,m)} als Ergebnis. Der Grund für diesen freiwilligen Informationsverzicht liegt darin, daß wir zunächst eine Form der Auswertung von Algebraausdrücken erhalten wollen, die bezüglich gesicherter Antworten der (einfachen) ω-Auswertung von erlaubten TRC-Anfragen entspricht. In Abschnitt 4.5 wird genauer auf das Äquivalenzproblem von TRC und Relationenalgebra bei ω-Relationen eingegangen.

Wir wollen die Auswertung eines Ausdrucks der Relationenalgebra sm-Auswertung nennen, wenn die oben eingeführten erweiterten Definitionen der Operationen verwendet werden.

Definition: Sei e ein beliebiger Ausdruck der Relationenalgebra vom Typ β zu einem DB-Schema σ, und sei z_ω ein ω-Zustand zu σ. Sei E die Menge aller Tupel, die die sm-Auswertung von e in z_ω liefert. Dann heißt E[STATUS=s][β] die sm-Antwort zu e in z_ω (in Zeichen: $\mathfrak{A}_{sm}(e,z_\omega)$).

Satz 4.4: Sei e ein beliebiger Ausdruck der Relationenalgebra vom Typ β zu einem DB-Schema σ, und sei z_ω ein ω-Zustand zu σ. Dann gilt $\mathfrak{A}_{sm}(e,z_\omega) \subseteq S_{uni}(e,z_\omega)$, d.h. eine sm-Antwort ist stets in der entsprechenden gesicherten uni-Antwort enthalten.

Beweis: Seien e ein beliebiger Algebraausdruck und t ein Tupel mit t(STATUS) = s, das im Ergebnis E von e bei Verwendung der erweiterten Operationen enthalten ist. e können wir in Form eines binären Baumes (Operatorbaumes) schreiben mit den Operationen als inneren Knoten und den Bezeichnern von Relationstypen als Blättern. Wir zeigen mit Induktion über die Tiefe i des Baumes die Gültigkeit der folgenden drei Eigenschaften und damit des Satzes:

1) $(E[STATUS=s])[\beta] \subseteq \mathcal{GT}(e,z_\omega)$

2) $(\forall t \in (E[STATUS=s])[\beta])(\forall t')(t' \trianglerighteq t \Rightarrow (\exists z \in POSS_c(z_\omega))(t' \in e(z)))$

3) $(\forall t)((\exists z \in POSS_c(z_\omega))(t \in e(z)) \Rightarrow (\exists t' \in E)(t \trianglerighteq t'|_\beta))$

i = 0: Dann ist e ein Bezeichner R eines Relationstyps. Alle Tupel in der durch R bezeichneten Relation des betrachteten Zustands haben den Wert s im Attribut STATUS und sind in der gesicherten uni-Antwort auf e enthalten. Damit sind 1 und 2 erfüllt. Tupel in möglichen Antworten auf e sind Überdeckungen der Tupel aus der mit R bezeichneten Relation, womit auch 3 gilt.

i → i+1: Wir müssen nur die Operationen Differenz, Selektion und Join betrachten, da die erweiterten Formen der Projektion und der Vereinigung wie die gewöhnlichen Operationen definiert sind und Vorkommen von ω wie Vorkommen gewöhnlicher Werte behandelt werden.

Seien e_1 und e_2 Ausdrücke mit Tiefe ≤ i, die Relationen R_1 und R_2 vom Typ δ bzw. γ als Ergebnisse haben, für die die drei Eigenschaften erfüllt sind.

a) Differenz: Sei $\beta = \gamma$; betrachte $e_1 \setminus_e e_2$.
Ein Tupel t ist nach Definition der erweiterten Differenz nur dann in $(R_1 \setminus_e R_2)[STATUS = s]$ enthalten, wenn es Element von $R_1[STATUS = s]$ ist und in R_2 kein mit ihm auf β verträgliches Tupel vorkommt. Sind zwei partielle Tupel nicht miteinander verträglich, dann sind auch alle Vervollständigungen dieser Tupel nicht miteinander verträglich. Daher gelten mit der Induktionsannahme die Eigenschaften 1 und 2.

Seien M_1, M_2 mögliche Relationen zu R_1 und R_2 und $t \in M_1 \setminus M_2$. Wegen $t \in M_1$ gibt es nach Induktionsannahme ein $u \in R_1$ mit $t \trianglerighteq u|_\beta$. Für u existiert nur dann kein Tupel u' mit $u'|_\beta = u|_\beta$ in $R_1 \setminus_e R_2$, wenn $Def(u) = \beta$ und es in R_2 ein Tupel v gibt mit $v|_\beta = u|_\beta$ und v(STATUS) = s. In diesem Fall gibt es aber nach Induktionsannahme auch in M_2 ein Tupel t' mit $t' = t = u|_\beta$ und t kann nicht Element von $M_1 \setminus M_2$ sein. Da t beliebig gewählt war, folgt die Gültigkeit der Eigenschaft 3 für $R_1 \setminus_e R_2$.

b) Selektion: Betrachte $e_1[vga]_e$.
Ein Tupel $t \in R_1$ mit t(STATUS) = s ist nach Anwendung der Selektionsoperation nur dann in $R_1[vga]_e[STATUS = s]$ enthalten, wenn t in den in vga angesprochenen Attributen total ist und vga erfüllt. Damit erfüllen auch alle Überdeckungen von t in den möglichen Relationen zu R_1 den Vergleichsausdruck, und 1 und 2 gelten für $e = e_1[vga]_e$ nach Induktionsannahme.

Ein Tupel $t \in R_1$ tritt weder mit s noch mit m als Wert in STATUS im Ergebnis auf, wenn t vga mit Sicherheit nicht erfüllt. Dies ist nur dann der Fall, wenn t in allen Attributen, die in vga angesprochen werden, definiert ist und vga nicht erfüllt wird. Damit erfüllen aber auch alle Überdeckungen von t in den möglichen Relationen zu R_1 diesen Ausdruck nicht und Eigenschaft 3 gilt ebenfalls nach Induktionsannahme.

c) Join: Betrachte $e_1 \bowtie e_2$.
Ein Tupel t über $\beta \cup \gamma$ ist nur dann in $(R_1 \bowtie_e R_2)[STATUS = s]$ enthalten, wenn $t|_\beta \in R_1[STATUS = s]$, $t|_\gamma \in R_2[STATUS = s]$ und $\beta \cap \gamma \subseteq Def(t)$. Wegen $\beta \cap \gamma \subseteq Def(t)$ werden auch in allen möglichen Relationen zu R_1 und R_2 die $t|_\beta$ bzw. $t|_\gamma$ überdeckenden Tupel über den gewöhnlichen Join miteinander verknüpft, weshalb mit der Induktionsannahme sowohl Eigenschaft 1 als auch Eigenschaft 2 für $R_1 \bowtie_e R_2$ erfüllt sind.

Betrachten wir nun ein beliebiges Tupel t im Ergebnis von $M_1 \bowtie M_2$ für zwei beliebige mögliche Relationen M_1 und M_2 zu R_1 und R_2. Nach Induktionsannahme gibt es Tupel $u \in R_1$ und $v \in R_2$ mit $t|_\beta \trianglerighteq u|_\beta$ und $t|_\gamma \trianglerighteq v|_\gamma$. t_1 und t_2 werden nur dann nicht durch $\bowtie_e$ zu einem Tupel verbunden, wenn sie auf $\beta \cap \gamma$ nicht verträglich miteinander sind. Nicht miteinander verträgliche Tupel haben in wenigstens einem Attribut zwei voneinander verschiedene definierte Werte. Vervollständigungen solcher Tupel werden aber auch mit dem gewöhnlichen Join nicht miteinander verbunden. Daher ist auch Eigenschaft 3 erfüllt. □

Betrachten wir nun etwas genauer die Information, die in Zwischenergebnissen bei der sm-Auswertung enthalten ist. Jedes gesicherte Tupel in einem solchen Zwischenergebnis ist ein gesichertes Antworttupel für den entsprechenden Teilausdruck betrachtet als Anfrage. Jedes mögliche Tupel ist in diesem Sinn ein Antworttupel, von dem nicht ausgeschlossen werden kann, daß es im Ergebnis enthalten

ist. Dies bedeutet nicht, daß ein mögliches Antworttupel auch in irgendeiner möglichen Antwort enthalten sein muß, d.h. es kann sich auch um ein Tupel handeln, das mit Sicherheit in keiner möglichen Antwort enthalten ist. Wir bezeichnen daher im folgenden diese Tupel als potentielle Tupel. Als Beispiel betrachte etwa den Ausdruck $R[A=3] \bowtie_e R[A\neq 3]$ mit $R = \{(1,\omega)\}$, für den wir $\{(1,\omega,m)\}$ als Ergebnis der sm-Auswertung erhalten.

In die sm-Antwort werden nur gesicherte Tupel übernommen; für die Auswertung der Operation auf äußerstem Niveau eines Ausdrucks würde es demnach genügen, nur gesicherte Tupel in das Ergebnis aufzunehmen. Es stellt sich allgemeiner die Frage, welche Information in den Operanden einer Operation genügt, um zu gewährleisten, daß alle Tupel im Ergebnis gesicherte (Antwort-) Tupel darstellen. Betrachten wir nacheinander die einzelnen Operationen.

Vereinigung: Diese Operation ist identisch mit der gewöhnlichen Form, so daß es genügt, wenn in den Operanden ebenfalls nur gesicherte Tupel enthalten sind.

Differenz: Da nur gesicherte Tupel aus dem ersten Operanden für die Antwort in Betracht kommen, genügt für diesen Operanden die Angabe der gesicherten Tupel. Jedes Tupel des zweiten Operanden, das mit einem Tupel des ersten Operanden verträglich ist, bewirkt, daß t nicht als gesichertes Tupel im Ergebnis auftritt. Die Unterscheidung in gesicherte und potentielle Tupel für den zweiten Operanden ist daher für die Bestimmung der gesicherten Tupel im Ergebnis ohne Bedeutung.

Projektion: Es gilt das gleiche wie für die Vereinigung.

Selektion: Da kein potentielles Tupel durch diese Operation zu einem gesicherten Tupel werden kann, genügt auch hier die Beschränkung auf die gesicherten Tupel des Operanden.

Join: Durch diese Operation können gesicherte Tupel nur aus Tupeln gebildet werden, die selbst gesichert sind. Es genügt also auch hier, für beide Operanden nur die gesicherten Tupel zu betrachten.

Aus den Betrachtungen ergeben sich somit folgende Bedingungen an die Operanden:

Für alle Operationen mit Ausnahme der Differenz genügt es für die Bestimmung der sm-Antwort, wenn in den Operanden nur die gesicherten Tupel bei der Auswertung betrachtet werden. Im Fall der Differenz ist es nicht notwendig, zwischen potentiellen und gesicherten Tupeln im zweiten Operanden zu unterscheiden. Alle Tupel werden hier benötigt und haben bezüglich der gesicherten Tupel im Resultat die gleiche Bedeutung.

Damit können wir eine zur sm-Auswertung äquivalente Methode zur Auswertung von Algebraausdrücken angeben, bei der kein zusätzliches Attribut notwendig ist. Stattdessen werden für die Operationen zwei Versionen zur Verfügung gestellt, um Operanden mit gesicherten bzw. potentiellen Tupeln erhalten zu können. Welche Version bei der Auswertung eines Ausdrucks zu benutzen ist, ergibt sich aus dem Aufbau des Ausdrucks. Wie wir gesehen haben, ist die Differenz die einzige Operation, bei der für einen Operanden alle Tupel, d.h. gesicherte und potentielle Tupel, benötigt werden. Hier findet also ein Wechsel statt für einen Operanden von der gesicherten zur möglichen Version. Dies bedeutet, daß bei der Berechnung eines solchen Operanden diejenige Version der zugehörigen Operation benutzt werden muß, die alle Tupel liefert, die sich möglicherweise qualifizieren.

Betrachten wir zunächst die beiden Versionen der Operationen, die *s-* bzw. *m-Version* genannt werden sollen, und durch eine entsprechende Indizierung des Operationssymbols kenntlich gemacht werden. Die Operationen sind auf gewöhnlichen ω-Relationen definiert, d.h. ohne das zusätzliche Attribut STATUS.

Seien R und U ω-Relationen über der gleichen Attributmenge α.

1) Vereinigung: $R \cup_s U =_{df} R \cup_m U =_{df} R \cup U$

2) Differenz: $R \setminus_s U =_{df} \{t \mid t \in R \wedge \neg(\exists t' \in U)(t' \Delta t)\}$

$R \setminus_m U =_{df} R \setminus \{t \mid Def(t) = \alpha \wedge t \in U\}$

3) Projektion: Sei $\beta \subseteq \alpha$; $R[\beta]_s =_{df} R[\beta]_m = R[\beta]$

4) Selektion: $R[vga]_s =_{df} \{t \mid t \in R \wedge t$ erfüllt vga mit Sicherheit$\}$

$R[vga]_m =_{df} R[vga]_s \cup \{t \mid t \in R \wedge t$ erfüllt vga möglicherweise$\}$

Seien nun U und V ω-Relationen über β bzw. γ mit $\beta,\gamma \subseteq \alpha$.

5) Join:

$$U \bowtie_s V =_{df} U[\beta \cap \gamma]_{total} \bowtie V[\beta \cap \gamma]_{total} \bowtie U \bowtie V$$

$$U \bowtie_m V = \{t \mid t \text{ ω-Tupel über } \beta \cup \gamma \wedge (\exists t' \in U)(\exists t'' \in V)\ (t'|_{\beta\cap\gamma} \Delta t''|_{\beta\cap\gamma} \wedge t|_{\beta\setminus\gamma} = t'|_{\beta\setminus\gamma} \wedge t|_{\gamma\setminus\beta} = t''|_{\gamma\setminus\beta} \wedge (t|_{\beta\cap\gamma} = \sup(t'|_{\beta\cap\gamma}, t''|_{\beta\cap\gamma})))\}$$

Für die Vereinigung und die Projektion stimmen die s- und die m-Version mit der gewöhnlichen Operation überein.

Bei der s-Version der Differenz werden nur diejenigen Tupel des ersten Operanden ins Ergebnis übernommen, zu denen es kein verträgliches Tupel im zweiten Operanden gibt. Für die m-Version sind dagegen alle Tupel des ersten Operanden zu übernehmen außer den Tupeln, die total sind und in beiden Operanden vorkommen.

Die s-Version der Selektion wählt alle Tupel des Operanden aus, die die angegebene Bedingung mit Sicherheit erfüllen, während die m-Version zusätzlich alle Tupel auswählt, die die Bedingung möglicherweise erfüllen.

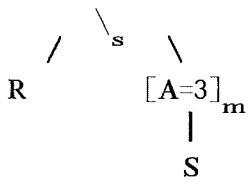
Beim s-Join werden alle Tupel der Operanden miteinander verbunden, die auf der gemeinsamen Attributmenge total definiert sind. Die m-Version ist analog zum erweiterten Join definiert. Ist die gemeinsame Attributmenge der beiden Operanden leer, dann ergibt sich auch hier das Kartesische Produkt als Sonderfall, da für $U,V \neq \Phi$ die Projektionen $U[\Phi]_{total}$ und $V[\Phi]_{total}$ jeweils eine nichtleere Menge liefern mit dem total undefinierten Tupel als einzigem Element.

Für die Auswertung von Ausdrücken können wir nun folgendes Verfahren angeben, das wir switch-Auswertung nennen wollen:

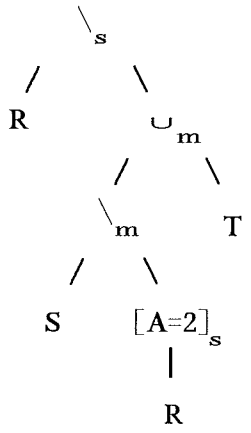
Sei e ein beliebiger Ausdruck der Relationenalgebra. Schreibe e in Form eines Operatorbaumes mit den Operationen als innere Knoten und den Relationsbezeichnern als Blätter. Führe dazu, falls nötig, e gemäß den üblichen Prioritäten in eine vollständig geklammerte Form über. Markiere die Knoten des Baumes von der Wurzel beginnend wie folgt:

- Die Wurzel wird mit "gesichert" markiert.
- Sei k ein beliebiger innerer Knoten des Baumes;
  falls k eine Vereinigung, eine Projektion, eine Selektion oder ein Join ist, markiere den oder die Nachfolger von k mit der gleichen Marke wie k;
  falls k eine Differenz ist, markiere den linken Nachfolger von k mit der gleichen Marke wie k und den rechten Nachfolger mit der anderen Marke.
- Werte den Ausdruck unter Verwendung derjenigen Versionen der Operationen aus, die durch den Baum bestimmt sind.

Beispiel 4.17: Für den Ausdruck R \ S[A=3] erhalten wir:



Für R \ ((S \ R[A=2]) ∪ T) ergibt sich:



Aus der Argumentation bei Einführung der Operationen folgt unmittelbar:

Satz 4.5: Die switch-Auswertung liefert die sm-Antwort als Ergebnis.

Eine naheliegende Idee ist die Verwendung der switch-Methode auch zur Berechnung *potentieller Antworten*, die aus potentiellen Antworttupeln bestehen, wie wir sie oben im Anschluß an Satz 4.4 diskutiert haben. Dazu muß lediglich die Markierung der Wurzel des Operatorbaumes mit m erfolgen; ansonsten kann in der gleichen Weise wie bei der switch-Auswertung verfahren werden.

Erkennen und Berücksichtigen identifizierbarer Vorkommen von ω

Die ω-Auswertung für TRC-Anfragen konnte auf einfache Weise erweitert werden, um Vergleiche eines Vorkommens von ω aus dem betrachteten DB-Zustand mit sich selbst berücksichtigen zu können. Dazu mußte nur die Information über die Belegungen von Variablen geeignet ausgenutzt werden, die sich wegen der Eindeutigkeit definierender Ausdrücke für jede Interpretation direkt ablesen läßt. Die Definition von Algebraoperationen erfolgt für beliebige Operandenmengen, deren Tupel im allgemeinen nicht mehr eindeutig einer Relation des betrachteten DB-Zustands zugeordnet werden können. Um auch hier in zweistelligen Operationen gleiche Vorkommen von ω erkennen zu können, muß eine solche Operation in Abhängigkeit vom gegebenen Ausdruck ausgeführt werden oder die Tupel in den Operanden müssen geeignete Information tragen.

Beispiel 4.18: Betrachte den folgenden TRC-Ausdruck zu einem geeigneten DB-Schema σ:

$(x)/\ (R\ x\ \wedge\ x.A > 3\ \vee\ S\ x) \wedge (\exists y)(R\ y\ \wedge\ x = y)$

Ist R x definierender Ausdruck, dann wird mit der erweiterten ω-Interpretation die Belegung von x und y mit dem gleichen Tupel berücksichtigt.
Betrachte einen entsprechenden Algebraausdruck:

$(R[A > 3] \cup S) \bowtie R$

Wegen des Funktionalitätsprinzips der Relationenalgebra ist vor Durchführung der Join-Operation zunächst der linke Operand auszuwerten. Im Ergebnis dieser Auswertung muß festgehalten werden, ob ein Tupel aus dem linken Operanden der Vereinigung oder aus dem rechten Operanden stammt oder in beiden Operanden vorkommt, um anschließend bei der Join-Operation analog zur erweiterten Form der ω-Auswertung identische Tupel beachten zu können. Die Analyse des Algebraausdrucks ohne Beachtung des gegebenen Zustands erlaubt es nicht, identische Tupel in den Operanden der Join-Operation identifizieren zu können, wie dies etwa (trivialerweise) für $(R \cup S) \bowtie R$ möglich ist.

Die Identität von ω-Tupeln und Vorkommen von ω in Operanden zweistelliger Operationen ist einfach festzustellen, wenn die Operanden durch Ausdrücke bestimmt sind, in denen nur Umbenennungen, Selektionen und Projektionen vorkommen, wobei für Projektionen gelten muß, daß keine Duplikatelimination auftreten kann (etwa dadurch gewährleistet, daß die Attributmenge, auf die projiziert wird, einen Schlüssel enthält). In diesen Fällen ist die Herkunft aller Tupel allein durch den Ausdruck eindeutig bestimmt. Daher können hier Definitionen für die Differenz (m-Version) und den Join (s-Version) verwendet werden, in denen Tupel mit gleichen Werten in allen Attributen identifiziert werden, wenn ihre Herkunft gleich ist.

Die Analyse von Anfragen im Hinblick auf eine solche Identifizierbarkeit von Tupeln und Vorkommen von ω kann in allgemeinerer Form erfolgen, was hier aber nicht weiter verfolgt werden soll.

Die Vereinigung ist eine Operation, die Kenntnisse über die Herkunft von Tupeln verwischen kann. Daher ist es naheliegend, Algebraausdrücke so umzuformen, daß Vereinigungen möglichst außen erfolgen. Dabei soll die Umformung einen bezüglich totaler Relationen äquivalenten Ausdruck erzeugen und so erfolgen, daß die sm-Antwort des erzeugten Ausdrucks die sm-Antwort des ursprünglichen Ausdrucks umfaßt.
Für den Ausdruck in Beispiel 4.18 kann folgender äquivalente Ausdruck gewählt werden:

$R[A > 3] \cup (S \bowtie R)$

Als Umformungsregeln bieten sich ähnliche Regeln für die einstelligen Operationen an, wie sie in Optimierungsstrategien für Algebraausdrücke verwendet werden ([Ull89], [EN94]): Führe Projektionen ohne Duplikatelimination und Selektionen so früh wie möglich durch, das heißt schiebe sie im Operatorbaum so weit wie möglich in Richtung der Blätter. Bei zweistelligen Operationen ist darauf zu achten, daß Operationen mit gleichem Operandentyp möglichst früh, das heißt so nah wie möglich an den Blättern durchgeführt werden.

Beispiel 4.19: Seien folgende Relationen gegeben:

R,S über {A,B,C} und T über {B,C,D};

dabei ist der jeweilige Schlüssel durch Unterstreichen der Attribute gekennzeichnet. Betrachte den folgenden Algebraausdruck:

$((((R \cup S[B = C]) \bowtie T)[A > 3])[A,B] \bowtie R)[C > 2]$

Durch Umformung erhalten wir:

$((R[A > 3] \bowtie T)[A,B] \overset{*}{\bowtie} R[C > 2]) \cup ((S[B = C \wedge A > 3] \bowtie T)[A,B] \bowtie R[C > 2])$

Bei Ausführung der mit * gekennzeichneten Join-Operation können Tupel in den Operanden mit gleichen Werten in allen Attributen identifiziert werden.

4.4.2 $\omega_{\{\}}$-Relationen und $\omega_{\{\}}$-Operationen

Wie oben schon erwähnt nutzt die sm-Auswertung, und damit auch die switch-Auswertung, nicht alle Informationen aus, die in den Operanden enthalten ist. Im folgenden wollen wir Möglichkeiten betrachten, wie diese Ausnutzung verbessert werden kann.

Bei den Definitionen der Operationen wurde kein Unterschied gemacht, ob ein Vorkommen von ω für einen einzelnen unbekannten Wert steht oder für eine mehrelementige Menge unbekannter Werte. Ein Vorkommen von ω in einem Tupel, das nicht durch Duplikatelimination aus mehreren anderen Tupeln entstanden ist, kann aber nur genau einen unbekannten Wert repräsentieren, da für die Relationen des Zustands die Vervollständigung als POSS-Funktion angenommen wird. Es bietet sich daher an, zwischen *gewöhnlichen* (*einelementigen*) und *mengenwertigen* Vorkommen von ω in Ergebnissen zu unterscheiden, wie wir dies schon bei der sub-Auswertung von TRC-Ausdrücken getan haben. Allerdings ergibt sich hier der Unterschied, daß von einem mengenwertigen Vorkommen von ω nicht bekannt ist, ob es eine Menge mit mehr als einem Element oder eine einelementige Menge repräsentiert. Da keine Verwechslung möglich ist, sei hier trotzdem das gleiche Symbol $\omega_{\{\}}$ für ein mengenwertiges Vorkommen von ω verwendet.

Bevor wir die geänderten Definitionen für die Operationen der Relationenalgebra angeben können, benötigen wir noch einige Festlegungen:

Für ein ω-Tupel t über einer Attributmenge α sei mit $t_{\{\}}$ dasjenige Tupel bezeichnet, das aus t entsteht, indem alle gewöhnlichen Vorkommen von ω durch mehrwertige Vorkommen ersetzt werden. Sollen die gewöhnlichen Vorkommen von ω nur in den Attributen einer Teilmenge β von α ersetzt werden, schreiben wir $t_{\{\}|\beta}$. Relationen, in denen eine solche Kennzeichnung erlaubt ist, bezeichnen wir als $\omega_{\{\}}$-Relationen.

Die Überdeckungsrelation ≥ und alle damit zusammenhängenden Definitionen seien auf Tupel mit mengenwertigen Vorkommen von ω erweitert, indem solche Vorkommen wie gewöhnliche Vorkommen von ω behandelt werden. In der gleichen Weise können auch die oben eingeführten s- und m-Versionen von Operationen sowie die gewöhnlichen Operationen auf $\omega_{\{\}}$-Relationen angewendet werden. Wir nutzen dies zur Vereinfachung von Definitionen.

Wir geben zunächst Definitionen für gesicherte Versionen der Algebraoperationen auf $\omega_{\{\}}$-Relationen an. Zur Unterscheidung von den oben eingeführten Operationen seien die Operationssymbole mit dem Index {} gekennzeichnet.

Seien R und U $\omega_{\{\}}$-Relationen über der gleichen Attributmenge α.

1) Vereinigung: $R \cup^{\{\}}_{s} U =_{df} \{t \mid t \in R \overset{a}{\vee} t \in U\} \cup$
$\{t_{\{\}} \mid t \in R \wedge (\exists t' \in U)(Def(t) = Def(t') \wedge t_{|Def(t)} = t'_{|Def(t')})\}$

Es ergibt sich gegenüber der gewöhnlichen Operation die Änderung, daß alle Vorkommen von ω in einem Ergebnistupel als mengenwertig gekennzeichnet werden, wenn es bei der gewöhnlichen Vereinigung ohne Kennzeichnung der mengenwertigen Vorkommen von ω zu einer Duplikatelimination kommen würde.

Beispiel: $\{(1,2), (1,\omega)\} \cup \{(\omega,2), (1,\omega)\} = \{(1,2), (1,\omega_{\{\}}), (\omega,2)\}$

2) Differenz: $R \setminus^{\{\}}_{s} U =_{df} \{t \mid t \in R \wedge \neg(\exists t' \in U)(t' \Delta t)\} \cup$
$\{t \mid (\exists t', t'' \in R \setminus_{m} U)(\neg(t' \Delta t'') \wedge$
$(\exists \beta \subseteq \alpha)(\beta = \{A \mid A \in Def(t') \cap Def(t'') \wedge t'(A) \neq t''(A)\} \wedge$
$t'_{|\alpha\setminus\beta} \Delta t''_{|\alpha\setminus\beta} \wedge t = (\inf(t',t''))_{\{\}|\beta} \wedge$
$(\exists u \in U)(u \Delta t' \wedge u \Delta t'' \wedge (\forall A \in \beta)(u(A) = \omega) \wedge$
$t', t'' \in R \setminus_{s} (U \setminus u)))))\}$

Zusätzlich zu den gesicherten Tupeln analog der s-Version der Differenz für gewöhnliche ω-Relationen werden dem Ergebnis noch Tupel $(\inf(t', t''))_{\{\}|\beta}$ hinzugefügt, wobei für t' und t'' gilt: t' und t'' unterscheiden sich mit Sicherheit genau in allen Attributen einer nicht leeren Menge β, auf der beide Tupel total definiert sind, und es gibt in U genau ein mit diesen Tupeln verträgliches Tupel, das bewirkt, daß beide Tupel (getrennt betrachtet) nicht in die gesicherte Antwort übernommen werden können, und das in allen Attributen der Menge β den Wert ω hat. Für jedes Paar möglicher Relationen zu R und U bleibt unter den angegebenen Bedingungen mindestens eins der beiden von t' und t'' induzierten Tupel im Ergebnis der Differenzbildung. Es kann also geschlossen werden, daß in jedem solchen Ergebnis ein von $(\inf(t',t''))_{\{\}|\beta}$ induziertes Tupel enthalten ist.

Beispiel: $\{(\omega,1,2), (\omega,1,3), (2,2,\omega_{\{\}}), (3,\omega,3)\} \setminus^{\{\}}_{s} \{(1,1,\omega), (\omega,2,3)\} = \{(\omega,1,\omega_{\{\}}), (\omega_{\{\}},\omega,\omega_{\{\}})\}$
Die Tupel $(\omega,1,2)$ und $(\omega,1,3)$ führen zu $(\omega,1,\omega_{\{\}})$ im Ergebnis mit u aus der Definition gleich $(1,1,\omega)$; $(2,2,\omega_{\{\}})$ und $(3,\omega,3)$ werden zu $(\omega_{\{\}},\omega,\omega_{\{\}})$ zusammengefaßt.

3) Projektion: Sei $\beta \subseteq \alpha$;
$R_{\{\}}[\beta]_{s} =_{df} \{t \mid (\exists t' \in R[\beta])(t = t' \wedge (\forall t'' \in R)(t'_{|\beta} = t''_{|\beta} \Rightarrow t' = t'') \vee$
$t = t'_{\{\}} \wedge (\exists t'' \in R)(t' \neq t'' \wedge Def(t'_{|\beta}) = Def(t''_{|\beta}) \wedge$
$t'_{|Def(t'_{|\beta})} = t''_{|Def(t'_{|\beta})}))\}$

Gibt es in R mehrere Tupel, die in ihren definierten Werten für Attribute aus β übereinstimmen, dann wird für diese Tupel in der Projektion auf β genau ein Tupel erzeugt, das mit ihnen in den definierten Werten übereinstimmt und in dem alle Vorkommen von ω als mengenwertig gekennzeichnet sind. Ansonsten ist die Wirkung der Operation wie die der gewöhnlichen Projektion.

Beispiel: $\{(1,2,\omega), (3,2,\omega), (2,3,\omega_{\{\}}), (3,3,\omega)\}[B,C] = \{(2,\omega_{\{\}}), (3,\omega_{\{\}})\}$

4), 5) Selektion und Join: Analog zur s-Version für gewöhnliche ω-Relationen, wobei Vorkommen von $\omega_{\{\}}$ wie Vorkommen von ω behandelt werden.

In den m-Versionen der Operationen auf ω-Relationen werden sowohl die Möglichkeit, daß ein Vorkommen von ω genau einen Wert repräsentiert als auch die Möglichkeit, daß ein solches Vorkommen eine mehrelementige Menge unbekannter Werte repräsentiert, berücksichtigt. Daher stimmen die m-Versionen der Operationen auf $\omega_{\{\}}$-Relationen mit den Definitionen der m-Versionen auf ω-Relationen überein, wenn Vorkommen von $\omega_{\{\}}$ wie Vorkommen von ω betrachtet werden.

In der gleichen Weise wie wir dies für ω-Relationen getan haben, kann nun für $\omega_{\{\}}$-Relationen eine switch-Auswertung festgelegt werden. Mit dieser Auswertungsform erhalten wir:

Satz 4.6: Die switch-Auswertung unter Verwendung von $\omega_{\{\}}$-Relationen liefert für alle Ausdrücke der Relationenalgebra stets ein Obermenge der gewöhnlichen switch-Auswertung als Ergebnis. Das Ergebnis ist immer eine Teilmenge der Menge der gesicherten Antworttupel, wenn Vorkommen von $\omega_{\{\}}$ wie Vorkommen von ω betrachtet werden.

Beweis: Die Betrachtung der Operationen zeigt, daß nur die gesicherte Version der Differenz zu einem echten Unterschied in bezug auf die switch-Auswertung führen kann. Alle anderen Operationen sind entweder gleich definiert wie die entsprechenden s- bzw. m-Versionen und behandeln Vorkommen von $\omega_{\{\}}$ wie Vorkommen von ω, oder sie dienen nur dem Vermerken von Duplikateliminationen (Vereinigung, Projektion).

Der Übergang von ω zu $\omega_{\{\}}$ bedeutet, daß mit ω nicht mehr das Wissen verknüpft ist, daß an der betreffenden Stelle genau ein Wert fehlt. Dieses Wissen wird nur bei Durchführung der s-Version der Differenz ausgenutzt. Gilt für diese Version der Differenz, daß für jedes Paar von $\omega_{\{\}}$-Relationen R und U das Ergebnis von $R \setminus_{\{\}s} U$ eine Teilmenge von $\mathcal{GT}$("R \ U",{R,U}) ist, dann folgt der Satz wegen der Identifizierung aller Vorkommen von ω und $\omega_{\{\}}$ bei den anderen Operationen aus den Sätzen 4.5 und 4.4. Die Erläuterung der s-Version der Differenz im Anschluß an ihre Definition liefert aber schon eine hinreichende Argumentation für die Gültigkeit von $R \setminus_{\{\}s} U \subseteq \mathcal{GT}$("R \ U",{R,U}). □

4.4.3 Komplexität der Operationen

Für totale Zustände sind Antworten auf Algebraanfragen genau wie bei TRC-Anfragen in polynomieller Zeit bestimmbar. Dies läßt sich einfach einsehen, wenn man die Operationsdefinitionen betrachtet. Mit einer direkten Umsetzung der Definitionen kann für jede Operation der Relationenalgebra schon ein Algorithmus mit quadratischer Laufzeit erhalten werden. Auch für einfache Operationen wie etwa die Vereinigung oder die Projektion gibt es aber keinen linearen Algorithmus, da eine eventuell notwendige Duplikatelimination zum Garantieren der Mengeneigenschaft von Ergebnissen berücksichtigt werden muß.

Für die Operationen auf den um das Attribut STATUS erweiterten ω-Relationen sowie für die s- und m-Versionen von Operationen kann einfach nachvollzogen werden, daß die Komplexität der Operationen die gleiche ist wie für gewöhnliche Algebraoperationen: Festzustellen, ob zwei Tupel verträglich miteinander sind, ist nicht schwieriger, als sie auf Gleichheit zu testen; die in den Definitionen der Differenz und des Join angegebenen Bedingungen sind für jedes Tupel des linken Operanden mit einem einzigen Durchlauf des zweiten Operanden überprüfbar.

Bei den Operationen auf $\omega_{\{\}}$-Relationen haben bis auf die s-Version der Differenz alle Operationen offensichtlich die gleichen Kosten wie im gewöhnlichen Fall. Die höheren Kosten der Differenz rühren daher, daß im ersten Operanden alle Tupelpaare betrachtet werden müssen und für jedes der beiden Tupel festgestellt werden muß, ob im zweiten Operanden genau ein Tupel existiert, das nach üblichem Vorgehen seine Entfernung aus dem gesicherten Ergebnis bewirkt. Die Kosten entsprechen demnach denjenigen für die gewöhnliche Differenz multipliziert mit $\binom{n}{2}$, n Anzahl der Tupel im ersten Operanden. Die üblichen Verbesserungen durch Einsatz von Sortieralgorithmen sind natürlich auch hier möglich.

4.5 Äquivalenz von TRC mit ω-Auswertung und Relationenalgebra mit switch-Auswertung

Im Standard-Relationenmodell mit totalen Zuständen gilt für die Relationenalgebra ohne Komplement, wie wir sie hier betrachten, daß es für jeden Ausdruck dieser Algebra zu einem DB-Schema σ eine entsprechende erlaubte TRC-Anfrage zu σ gibt, die für jeden Zustand den Wert des Ausdrucks als Antwort hat. Die Umkehrung gilt ebenfalls. Dabei sind für den TRC die Antworten unter der CWA eindeutig über die Standard-Interpretationsvorschrift der Prädikatenlogik bestimmt.

Bemerkung: Für diese Äquivalenz ist es nicht ausreichend, als TRC-Ausdruck nur R-beschränkte Ausdrücke oder Ausdrücke der Form, wie sie in [Cod72b] angegeben wurde, zuzulassen: Zu dem Algebraausdruck $R \cup S[\beta]$ mit $\beta \subseteq Attr(R) \subsetneq Attr(S)$ existiert z.B. kein äquivalenter R-beschränkter Ausdruck. Begründung: Da das Ergebnis des Ausdrucks vom Typ β ist und alle Tupel aus R enthalten muß, ist die freie Variable einer äquivalenten TRC-Anfrage an R zu binden. Eine zusätzliche Bindung an S, etwa wie in $R\ x \vee S\ x$, ist nicht möglich, da R und S verschiedenen Typ haben. Damit sind aber alle Tupel aus $S[\beta]$ über x nicht ansprechbar. Die in [Cod72b] aufgestellte Äquivalenzbehauptung ist daher in einer Richtung (die in [Cod72b] auch nicht bewiesen wird) nur mit geeigneter Erweiterung der Definition R-beschränkter TRC-Ausdrücke korrekt (was im Anhang von [Cod72b] auch erwähnt wird).

Es stellt sich die Frage, für welche Operationsdefinitionen und Interpretationsvorschriften Äquivalenzen von Algebra- und TRC-Ausdrücken im Fall partieller Zustände gelten. Ein erstes Problem ergibt sich daraus, daß Äquivalenzen von Ausdrücken in einer Sprache, die im Fall totaler Zustände und Standarddefinitionen bzw. Standardvorschriften gelten, nicht notwendigerweise auch Äquivalenzen unter den eingeführten Interpretationsvorschriften und Operationsdefinitionen für partielle Zustände sein müssen.

Beispiel 4.20: Seien $R = \{(1,4)\}$ und $S = \{(1,\omega)\}$ ω-Relationen über {A,B}. Betrachte die beiden folgenden TRC-Anfragen:

$(x)/\ R\ x \wedge \neg(\exists y)(S\ y \wedge y.B = 3 \wedge x.A = y.A \wedge x.B = y.B)$

$(x)/\ R\ x \wedge \neg(\exists y)(S\ y \wedge y.B = 3 \wedge x.A = y.A \wedge x.B = 3)$

Der Übergang von der ersten zur zweiten Anfrage wäre etwa als ein Optimierungsschritt denkbar. Mit der ω-Auswertung (gleich welcher Version) wird für die erste Anfrage die leere Antwort erhalten, da der quantifizierte Teilausdruck zu

unbekannt ausgewertet wird. Für die zweite Anfrage ergibt sich dagegen R als Antwort, da der Teilausdruck hier den Wert *falsch* erhält. R ist auch die gesicherte uni-Antwort auf beide Anfragen. Die Anfragen sind für totale Zustände äquivalent, unter den eingeführten Interpretationsvorschriften im allgemeinen aber nicht für ω-Zustände, wie das Beispiel zeigt.

Der erste der beiden folgenden Algebraausdrücke entspricht der ersten TRC-Anfrage aus obigem Beispiel, der zweite Ausdruck ist für totale Zustände zu ihm äquivalent:

$R \setminus S[B=3]$

$R[B\neq 3] \cup (R[B=3] \setminus S)$

Die sm-Antwort zum ersten Ausdruck ist leer (wegen der nicht erfolgenden Berücksichtigung der Information aus der Selektionsbedingung). Für den zweiten Ausdruck erhalten wir dagegen R als Antwort.

Das nächste Beispiel macht ein grundsätzliches Problem deutlich, warum die ω-Auswertung für eine TRC-Anfrage zu einem anderen Ergebnis führen kann als die Auswertung einer im gewöhnlichen Fall äquivalenten Algebraanfrage unter Verwendung der erweiterten Operationsdefinitionen.

Beispiel 4.21: Seien $R = \{(1,4)\}$, $S = \{(1,\omega)\}$ und $T = \{(1,3)\}$ ω-Relationen über {A,B}. Für den Ausdruck $R \setminus (S \bowtie T)$ erhalten wir mit der sm- bzw. switch-Auswertung R als Ergebnis. Durch die $\leq$-sup-Bildung bei der m-Version des Join wird vermerkt, daß nur das Tupel (1,3) als Ergebnis von $S \bowtie T$ möglich ist. Ein derartiges "Merken" gibt es bei der ω-Auswertung nicht. Dies kann dazu führen, daß die syntaktische Struktur eines entsprechenden TRC-Ausdrucks in Abhängigkeit vom DB-Zustand bestimmt, ob der TRC-Ausdruck mit der ω-Auswertung das gleiche Ergebnis wie der Algebraausdruck mit geeigneten Operationsdefinitionen liefert. Betrachten wir dazu die beiden Ausdrücke

$(x)/\ R\ x \wedge \neg (\exists y)(S\ y \wedge (\exists z)(T\ z \wedge y.A = z.A \wedge y.B = z.B \wedge y.A = x.A \wedge y.B = x.B))$

und

$(x)/\ R\ x \wedge \neg (\exists y)(S\ y \wedge (\exists z)(T\ z \wedge y.A = z.A \wedge y.B = z.B \wedge z.A = x.A \wedge z.B = x.B)).$

Beide Anfragen sind im gewöhnlichen Fall äquivalent zu $R \setminus (S \bowtie T)$. Der einzige Unterschied zwischen den Anfragen besteht darin, daß der Vergleich mit x einmal über y und einmal über z ausgedrückt wird. Für die erste Anfrage erhalten wir mit der ω-Auswertung die leere Antwort: Die Konjunktion liefert für die einzig mögliche Belegung der Variablen *unbekannt*. Die zweite Anfrage hat dagegen R als Antwort, da die Konjunktion wegen des letzten Vergleichsausdrucks *falsch* als Antwort liefert.

Die "Merkmöglichkeit" im Join führt auch dazu, daß die Join-Operation im allgemeinen nicht mehr äquivalent formuliert werden kann durch das Kartesische Produkt mit geeigneter Umbenennung und anschließender Selektion und Projektion. So erhält man für den

$R \setminus (S \bowtie T)$

entsprechenden Ausdruck

$R \setminus (((S \times T[A \rightarrow A',\ B \rightarrow B'])[A = A' \wedge B = B'])[A,B])$

mit der sm- bzw. switch-Auswertung die leere Antwort.

Die Beispiele zeigen, daß für einen Vergleich der ω-Auswertung mit erweiterten Algebraoperationen auf die Form der wechselseitigen Übersetzung von Anfragen zu achten ist bzw. daß für den Join im allgemeinen keine äquivalente Formulierung im TRC gefunden werden kann. Wir beschränken die folgende Äquivalenzbetrachtung daher auf Ausdrücke, in denen der Join nur in der Form, die dem Kartesischen Produkt entspricht (der Schnitt der Attributmengen der Operanden ist leer), erlaubt ist. Zur Vereinfachung des Vergleichs zeigen wir zunächst, daß die Idee der switch-Auswertung auch bei der ω-Auswertung angewendet werden kann.

Definition: Zu einem beliebigen erlaubten TRC-Ausdruck $\mathfrak{a}$ mit den freien Variablen $x_1,....,x_n$ sei die potentielle Antwort definiert als die Menge aller Belegungen von $x_1,....,x_n$, mit denen die ω-Auswertung für $\mathfrak{a}$ den Wert *wahr* oder *unbekannt* liefert. Hat $\mathfrak{a}$ keine freien Variablen, dann bedeutet dies, daß $\mathfrak{a}$ auch dann der Wert *wahr* zugeordnet wird, wenn die ω-Auswertung für $\mathfrak{a}$ den Wert *unbekannt* ergibt.

Potentielle Antworten unterscheiden sich von möglichen Antworten dadurch, daß sie nicht als Antwort in einem möglichen Zustand erhalten werden müssen (vergleiche mit potentiellen Tupeln und Antworten bei Algebraausdrücken).

Die gleiche Rolle, die die Differenz in Algebraausdrücken bei der switch-Auswertung spielt, spielt die Negation in TRC-Ausdrücken: Für $\mathfrak{a}$ selbst wird die gesicherte Antwort berechnet; tritt in $\mathfrak{a}$ ein negierter Teilausdruck auf, der nicht selbst Teil eines negierten Ausdrucks ist, dann wird für diesen Teilausdruck die potentielle Antwort berechnet. Weitere Negationen bewirken wieder einen Wechsel zur Berechnung einer gesicherten (Teil)Antwort usw. Sei die so geänderte ω-Auswertung als switch-ω-Auswertung bezeichnet.

Beispiel 4.22: Betrachten wir die beiden TRC-Anfragen aus Beispiel 4.21.
Für die erste Anfrage muß für den quantifizierten Teilausdruck wegen der Negation die potentielle Antwort (mit (1,4) als einziger zu betrachtender Belegung von x) bestimmt werden. Sie ist nach Festlegung *wahr*, da die ω-Auswertung *unbekannt* ergibt.
Für die zweite Anfrage ändert sich beim Übergang zur switch-ω-Auswertung nichts, da der quantifizierte Teilausdruck mit der ω-Auswertung zu *falsch* ausgewertet wird.
Wir erhalten demnach für beide Anfragen mit der ω-Auswertung die gleiche Antwort wie mit der switch-ω-Auswertung.

Satz 4.7: Die switch-ω-Auswertung liefert für jede TRC-Anfrage die gleiche Antwort wie die ω-Auswertung.

Beweis: Sei ein beliebiger Ausdruck $\mathfrak{a}$ gegeben, in dem für einen Teilausdruck $\neg\mathfrak{b}$ nach der beschriebenen Methode die potentielle Antwort für $\mathfrak{b}$ zu ermitteln ist. Sei $\mathfrak{a}$ gemäß der ω-Auswertung berechnet mit dem Unterschied, daß für $\mathfrak{b}$ immer die potentielle Antwort für die jeweilige Belegung der freien Variablen von $\mathfrak{b}$ ermittelt wird, d.h. es erfüllen auch diejenigen Belegungen der freien Variablen von $\mathfrak{b}$ den Teilausdruck $\mathfrak{b}$, für die *unbekannt* bei der Auswertung von $\mathfrak{b}$ erhalten wird.
Behauptung: Durch diese Abänderung wird die ω-Antwort auf $\mathfrak{a}$ nicht verändert.
Beweis der Behauptung: Es sind diejenigen Belegungen von $\mathfrak{b}$ zu betrachten, für die *unbekannt* bei der Auswertung von $\mathfrak{b}$ erhalten wird. Sie werden in der gleichen Weise behandelt wie Belegungen, für die *wahr* als Wert erhalten wird. Für $\neg\mathfrak{b}$

wird damit aber nur dann *wahr* für eine Belegung der freien Variablen von $\mathfrak{b}$ erhalten, wenn $\mathfrak{b}$ mit dieser Belegung den Wert *falsch* hat. Es ändert sich demnach nichts an der Menge der Belegungen, die $\neg\,\mathfrak{b}$ erfüllen. Wegen der angenommenen Negationsstruktur, die $\mathfrak{b}$ umgibt, können die zusätzlichen Belegungen somit keine Veränderung der ω-Antwort auf $\mathfrak{a}$ bewirken.

An $\mathfrak{a}$ und den Teilausdruck $\mathfrak{b}$ wurden keine weiteren Voraussetzungen gestellt, weshalb der Satz mit der Behauptung bewiesen ist. □

Nach diesen Vorbereitungen können wir den folgenden Äquivalenzsatz beweisen:

Satz 4.8: Zu jeder Anfrage $\mathfrak{e}$ der Relationenalgebra (ohne Komplement und nur mit dem Join in der speziellen Form des kartesischen Produkts) zu einem DB-Schema σ gibt es eine erlaubte TRC-Anfrage, die für jeden ω-Zustand zu σ mit der ω-Auswertung die gleiche Antwort liefert wie $\mathfrak{e}$ bei Anwendung der switch-Auswertung und umgekehrt.

Beweis: Wir zeigen die Richtung Algebra $\rightarrow$ TRC durch Induktion über den Aufbau von Algebraausdrücken, wobei wir uns an der Beweisführung für die Äquivalenz im gewöhnlichen Fall in [KK93] orientieren.

Für jede Operation ist zu zeigen, daß die s-Version und die m-Version jeweils das gleiche Ergebnis liefern wie die Bestimmung der gesicherten bzw. der potentiellen Antwort für die entsprechende TRC-Anfrage. Wegen der an der syntaktischen Struktur von Algebraausdrücken orientierten Form des Beweises sowie wegen der Möglichkeit zur eindeutigen Zuordnung von Differenzbildungen in Algebraausdrücken zu Negationen in den Qualifikationsausdrücken der entsprechenden TRC-Anfragen ergibt sich damit auch die betrachtete Äquivalenz insgesamt.

Die Induktion erfolgt über die Schachtelungstiefe von Ausdrücken, für die wir vollständige Klammerung voraussetzen. Sei $\mathfrak{e}$ ein beliebiger Algebraausdruck der Tiefe i.

i = 0: Es ist kein Operator vorhanden. Eine Anfrage hat dann die Form

R oder {(A:a)},

wobei R der Bezeichner eines Relationstyps des betrachteten DB-Schemas und A ein Attribut aus der Attributmenge des Schemas ist. Die entsprechenden TRC-Anfragen haben die Form

(x)/ R x mit typ(x) = Attr(R) bzw. (x)/ x.A = a mit typ(x) = {A}.

Da es in ihnen nur einen Bereichsausdruck bzw. einen Vergleichsausdruck gibt, sind beide Ausdrücke jeweils definierende Ausdrücke. Damit qualifiziert sich jedes Tupel aus der mit R bezeichneten Relation bzw. das Tupel (a) für die gesicherte und für die potentielle Antwort. Da (außer der Identität) keine Operationen in den Algebraausdrücken vorkommen, müssen s- und m-Versionen nicht unterschieden werden. Die Antwort ist identisch mit derjenigen auf die TRC-Anfragen.

Induktionsannahme: Für alle Algebraausdrücke mit Tiefe h $\leq$ i gelte, daß das gesicherte und das potentielle Ergebnis mit der gesicherten bzw. potentiellen Antwort der entsprechenden TRC-Anfrage übereinstimmen.
Wir schreiben $\mathfrak{e} \sim \mathfrak{a}$ für Algebraausdrücke $\mathfrak{e}$ und TRC-Anfragen $\mathfrak{a}$, wenn $\mathfrak{a}$ eine zu $\mathfrak{e}$ gehörende Anfrage ist.

Schritt von i auf i+1: Betrachte eine Anfrage $\mathfrak{e}$ mit Tiefe i+1. Wir diskutieren nacheinander die einzelnen Operationen als äußere Operationen von $\mathfrak{e}$.

1) Vereinigung: Sei $e = e_1 \cup e_2$, e_1 und e_2 mit Tiefe $\leq i$.

Es gelte $e_1 \sim (x)/\ \Phi_1(x)$, $e_2 \sim (y)/\ \Phi_2(y)$ mit typ(x) = typ(y) = ζ. Dann gilt:

$e \sim (z)/\ (\exists x)(\Phi_1(x) \wedge x.\zeta = z.\zeta) \vee (\exists y)(\Phi_2(y) \wedge y.\zeta = z.\zeta)$

mit typ(z) = ζ.

Jedes Attribut von z ist über genau einen Vergleichsausdruck positiv beschränkt; damit sind in der Antwort genau alle Tupel enthalten, die in der Antwort auf e_1 (kurz auch: in e_1) oder e_2 enthalten sind. Die s- und die m-Version der Vereinigung unterscheiden sich nicht und ergeben nach Definition ebenfalls dieses Ergebnis.

2) Differenz: Sei $e = e_1 \setminus e_2$, e_1 und e_2 mit Tiefe $\leq i$.

Es gelte wieder $e_1 \sim (x)/\ \Phi_1(x)$, $e_2 \sim (y)/\ \Phi_2(y)$ mit typ(x) = typ(y) = β.

Der entsprechende erlaubte TRC-Ausdruck lautet dann:

$(x)/\ \Phi_1(x) \wedge \neg (\exists y)(\Phi_2(y) \wedge x.\beta = y.\beta)$.

Betrachten wir zunächst die s-Version der Differenz. Für e_2 liegt gemäß switch-Auswertung das potentielle Ergebnis vor. In das Ergebnis von $e_1 \setminus_s e_2$ werden nach Definition nur diejenigen Tupel aus e_1 aufgenommen, zu denen in e_2 kein verträgliches Tupel existiert. Im quantifizierten Teilausdruck der TRC-Anfrage führen beim Vergleich x.β = y.β alle Belegungen von x und y, die nicht verträglich miteinander sind, zu *falsch*. Dies ist unabhängig davon, ob x.β = y.β definierend oder gewöhnlich ist. Gibt es zu einem Tupel t in e_1 kein verträgliches Tupel im potentiellen Ergebnis von e_2, dann gibt es nach Induktionsannahme auch keine Belegung für y, so daß der quantifizierte Teilausdruck mit t als Belegung von x zu *wahr* ausgewertet wird. Wegen der anschließenden Negation ist daher mit der Induktionsannahme das Ergebnis von $e_1 \setminus_s e_2$ in der ω-Antwort auf die TRC-Anfrage enthalten. Es bleibt zu zeigen, daß sich keine weiteren Belegungen von x qualifizieren. Weitere Belegungen können sich nur dann qualifizieren, wenn der quantifizierte Teilausdruck (ohne Negation) mit einer solchen Belegung für jede Belegung von y den Wert *falsch* hat.

Seien eine beliebige Belegung t von x mit $\Phi_1(t)$ ist *wahr* gegeben. t muß mit wenigstens einer Belegung t' von y, die zur potentiellen Antwort auf (y)/ $\Phi_2(y)$ gehört, verträglich sein, um nicht zu den oben betrachteten Belegungen zu gehören. Ist x.β = y.β definierend und gehört t' zur gesicherten Antwort auf (y)/ $\Phi_2(y)$, dann muß t' total sein, da y in $\Phi_2(y)$ über mindestens einen gewöhnlichen Bereichs- oder Vergleichsausdruck beschränkt ist und $\Phi_2(t')$ den Wert *wahr* hat. t und t' sind damit identisch und der quantifizierte Teilausdruck hat den Wert *wahr*. Gehört t' nicht zur gesicherten Antwort auf (y)/ $\Phi_2(y)$, dann ist der Wert von $\Phi_2(t')$ *unbekannt*.

Ist x.β = y.β gewöhnlich, dann kann t oder t' nicht total sein, da sonst aus der Verträglichkeit von t und t' die Gleichheit von t und t' folgt. Damit hat aber t.β = t'.β den Wert *unbekannt*.

Es gelingt also stets, unter den genannten Voraussetzungen eine Belegung von y zu finden, so daß für die Konjunktion im quantifizierten Teilausdruck der Wert *wahr* oder *unbekannt* erhalten wird.

Für $e_1 \setminus_m e_2$ werden neben den Tupeln, die in $e_1 \setminus_s e_2$ enthalten sind, noch Tupel aus e_1 ins Ergebnis aufgenommen, die nicht total sind oder die total sind und zu denen es kein identisches Tupel in e_2 gibt. Für die Bestimmung der poten-

tiellen Antwort des TRC-Ausdrucks ist für den Existenz-quantifizierten Teilausdruck die gesicherte Antwort für jede Belegung von x zu bestimmen. Damit qualifizieren sich nur diejenigen Belegungen von x, die total definiert sind, Φ_1 erfüllen und für die es eine identische Belegung von y gibt, die sich gemäß Φ_2 qualifiziert. Die Negation des Teilausdrucks sorgt dafür, daß sich genau diese Tupel für die Antwort nicht qualifizieren. Wir erhalten also das gleiche Ergebnis wie für die Differenz.

3) Projektion: Sei $\beta \subseteq \alpha$ und $e = e_1[\beta]$, e_1 mit Tiefe i.
Es gelte wieder $e_1 \sim (x)/\ \Phi_1(x)$.
Der entsprechende TRC-Ausdruck lautet

$(x.\beta)/\ \Phi_1(x)$.

Dieser Ausdruck unterscheidet sich von dem üblicherweise gewählten Ausdruck $(y)/\ (\exists x)(\Phi_1(x) \wedge x.\beta = y.\beta)$ (siehe [KK93]) und bedingt einige Änderungen beim induktiven Einsetzungsprozeß. Der Grund hierfür liegt darin, daß $(y)/\ (\exists x)(\Phi_1(x) \wedge x.\beta = y.\beta)$ im Fall von ω-Relationen nur bezüglich der potentiellen Antwort die Projektion widerspiegelt. Bei der Bildung der gesicherten Antwort gehen dagegen alle auf β nicht totalen Tupel verloren. Die gewählte Abänderung hat folgende Auswirkung auf den induktiven Einsetzungsprozeß: Tritt eine Projektion als äußerste Operation eines Ausdrucks auf, ist keine Änderung vorzunehmen, sondern $(x.\beta)$ stellt dann genau die Zielliste der durch den Einsetzungsprozeß erzeugten TRC-Anfrage dar. Kommt eine Projektion im Innern eines Ausdrucks vor, dann hat dies nur Auswirkungen auf den Typ einer Variablen im Ersetzungsschritt, der dem die Projektion betreffenden Schritt folgt sowie auf das Belegen dieser Variablen. Wir zeigen die Auswirkung für die Operation Vereinigung.
Sei $e = e_1 \cup e_2$. Es sei $(x.\beta)/\ \Phi_1(x)$ der entsprechende TRC-Ausdruck zu e_1 und $(y)/\ \Phi_2(y)$ der entsprechende TRC-Ausdruck zu e_2, wobei $typ(y) = \beta$ gelten soll. Dann ist

$(z)/\ (\exists x)(\Phi_1(x) \wedge x.\beta = z.\beta) \vee (\exists y)(\Phi_2(y) \wedge y.\beta = z.\beta)$ mit $typ(z) = \beta$

der entsprechende TRC-Ausdruck zu $e_1 \cup e_2$.

4) Selektion: Sei $e = e_1[A \ominus c]$ oder $e = e_1[A \ominus B]$, e_1 mit Tiefe i.
Es gelte $e_1 \sim (x)/\ \Phi_1(x)$. Dann ordnen wir e die folgende TRC-Anfrage zu:

$(x)/\ \Phi_1(x) \wedge x.A \ominus c$ bzw. $(x)/\ \Phi_1(x) \wedge x.A \ominus x.B$

Bei der s-Version der Selektion qualifizieren sich alle Tupel des Operanden, die den Vergleichsausdruck mit Sicherheit erfüllen. Dies sind mit der Induktionsannahme genau die Tupel, die in der ω-Antwort auf $(x)/\ \Phi_1(x) \wedge x.A \ominus c$ bzw. $(x)/\ \Phi_1(x) \wedge x.A \ominus x.B$ enthalten sind.
Für die m-Version und die potentielle Antwort gilt entsprechendes.

5) Kartesisches Produkt: Sei $e = e_1 \times e_2$; e_1 und e_2 mit Tiefe ≤ i.

Seien $e_1 \sim (x)/\ \Phi_1(x)$ und $e_2 \sim (y)/\ \Phi_2(y)$ mit $typ(x) = \beta$, $typ(y) = \gamma$ und $\beta \cap \gamma = \Phi$.
Dann gilt:

$e \sim (z)/\ (\exists x)(\exists y)(\Phi_1(x) \wedge \Phi_2(y) \wedge z.\beta = x.\beta \wedge z.\gamma = y.\gamma)$ mit $typ(z) = \beta \cup \gamma$.

Betrachten wir die Definition der s- und der m-Version des Join, dann stellen wir fest, daß sich beide Versionen für den Sonderfall des kartesischen Produktes

nicht unterscheiden. Als Ergebnis erhalten wir in beiden Fällen alle Tupel, die sich aus den Verknüpfungen aller Tupel in den Operanden ergeben.
Da in der TRC-Anfrage alle Vergleichsausdrücke mit z definierend sein müssen, werden in der ω-Antwort alle Tupel aus den ω-Antworten zu (x)/ $\Phi_1(x)$ und (y)/ $\Phi_2(y)$ miteinander verknüpft. Entsprechendes gilt für die potentielle Antwort auf die TRC-Anfrage. Die Übereinstimmung dieser Antworten mit den Ergebnissen, die die s- bzw. m-Version des Join liefern, folgt daher mit der Induktionsannahme.

6) Umbenennung: Sei $\mathfrak{e} = \mathfrak{e}_1[A_1 \rightarrow B_1,....,A_k \rightarrow B_k]$, $\mathfrak{e}_1$ mit Tiefe i.
Sei wiederum $\mathfrak{e}_1 \sim$ (x)/ $\Phi_1(x)$ mit typ(x) = β. Dann gilt:

$\mathfrak{e} \sim$ (y)/ $(\exists x)(\Phi_1(x) \wedge y.B_1 = x.A_1 \wedge .... \wedge y.B_k = x.A_k \wedge y.C_1 = x.C_1 \wedge .... \wedge y.C_\ell = x.C_\ell)$

mit $\{C_1,....,C_\ell\} = \beta \setminus \{A_1,....,A_k\}$ und typ(y) = $\{B_1,....,B_k\} \cup \{C_1,....,C_\ell\}$

Da alle Vergleichsausdrücke mit y definierend sind, erfolgt eine Übernahme aller Tupel, die sich für die entsprechende Antwort auf (x)/ $\Phi_1(x)$ qualifizieren, in das Ergebnis. Die Wirkung der Anfrage ist demnach nur eine Typänderung der Antworttupel, was der Wirkung der Umbenennungsoperation entspricht.

Der Beweis der Umkehrrichtung TRC → Relationenalgebra kann ebenfalls auf Transformationsvorschriften für den gewöhnlichen Fall zurückgreifen, in denen weder der Join noch die Division als Operationen verwendet werden. Eine solche Vorschrift ist etwa aus der in [KK93] angegebenen Vorschrift direkt abzuleiten. Da im Beweis gegenüber der gezeigten Richtung keine wesentlich andere Argumentation notwendig ist, soll er entfallen. □

5 Enge Ausdehnungen von ω-Relationen

Die enge Ausdehnung stellt als POSS-Funktion eine Verallgemeinerung der im vorangehenden Kapitel betrachteten Vervollständigung dar: Für jede ω-Relation enthält die Menge der möglichen Relationen unter der engen Ausdehnung als POSS-Funktion die Menge der möglichen Relationen unter der Vervollständigung als POSS-Funktion. Wie wir zeigen werden, führt diese Verallgemeinerung dazu, daß gesicherte Antworten gleich welcher der in Kapitel 3 vorgestellten Formen selbst im Fall einfacher DB-Schemata für beliebige ω-Zustände nicht mehr berechenbar sind. Bevor wir zum Beweis dieses Resultates kommen, betrachten wir zunächst einen Vorschlag, durch den die Probleme mit dem Ansatz von E.F. Codd, den wir in Kapitel 3 diskutiert haben, vermieden werden. Dieser Vorschlag ändert allerdings durch den Übergang von der Vervollständigung zur engen Ausdehnung als Z-POSS-Funktion die Semantik von ω-Zuständen ab und kann daher nur eingeschränkt als "Reparatur" des Codd'schen Ansatzes gesehen werden.

In weiteren Abschnitten dieses Kapitels geben wir für den TRC und die Relationenalgebra Verfahren zur Bestimmung gesicherter Antworten an. Wir erhalten sie durch entsprechende Anpassungen der Verfahren aus Kapitel 4 an die geänderte POSS-Funktion.

5.1 Der Vorschlag von J. Biskup

Zur Vermeidung der Probleme, die beim Vorschlag von E.F. Codd auftreten, hat J. Biskup in [Bis83] den Übergang zur engen Ausdehnung als POSS-Funktion für Zustände und Antworten sowie eine Erweiterung von Relationstypen vorgeschlagen. Die Erweiterung der Relationstypen besteht in der Hinzunahme von Wertebereichsangaben in die Relationen und einem zusätzlichen Attribut mit zweielementigem Wertebereich. Dieses zusätzliche Attribut, das für alle Relationstypen gleich ist, bietet die Möglichkeit zur Unterscheidung von *gesicherten* und *möglichen* Tupeln. Es werden nur Operationen der Relationenalgebra untersucht; Kalkülanfragen werden nicht betrachtet.

Die Hinzunahme der Wertebereichsangaben erfolgt, um Eigenschaften von Operationen der Relationenalgebra untersuchen zu können. Zusammen mit dem Relationstyp sind für jedes Attribut Angaben zu machen, die eine endliche Teilmenge der Menge aller Konstanten als möglichen Wertebereich festlegen. Wegen dieser Beschränkung ist die Anzahl möglicher Relationen zu einer ω-Relation stets endlich. Wertebereichsangaben werden bei den Definitionen der Operationen allerdings nicht ausgenutzt, um Komplexitätsprobleme im Zusammenhang mit gewissen Vollständigkeitsanforderungen zu vermeiden. Da nur einzelne elementare Operationen und keine komplexen Ausdrücke betrachtet werden, spielt die im nächsten Abschnitt gezeigte Nicht-Berechenbarkeit gesicherter Antworten trotzdem keine Rolle.

Die Unterscheidung von Tupeln in gesicherte und mögliche Tupel haben wir schon in Abschnitt 4.4.1 betrachtet. Dort haben wir von potentiellen statt von möglichen Tupeln gesprochen, um Verwechslungen mit Tupeln von möglichen Antworten zu vermeiden. Diese Sprechweise wollen wir hier beibehalten. Aus Satz 4.5 (S. 90) folgt, daß die Unterscheidung von gesicherten und potentiellen Tupeln in Relationen nicht notwendig ist für die Berechnung gesicherter Antworten, wenn

geeignete Operationsdefinitionen in Abhängigkeit vom Aufbau eines Ausdrucks verwendet werden. Das gleiche gilt für potentielle Antworten.

Einen Vorteil bietet die Unterscheidung in gesicherte und potentielle Tupel für die Bewertung von Operationsdefinitionen im Hinblick auf die Verwertung der Information in den Operanden. Dazu werden in [Bis83] zwei Eigenschaften eingeführt: *hinreichend* (*adequate*) und *beschränkt* (*restricted*). Wir haben diese Eigenschaften schon bei der Definition gesicherter min/max-Antworten erwähnt.

Eine Operation auf ω-Relationen hat die Eigenschaft hinreichend, wenn für alle Operanden jede mögliche Antwort als POSS-Relation des Ergebnisses darstellbar ist. Eine Operation hat die Eigenschaft beschränkt, wenn für alle Operanden das Ergebnis stets minimal ist in dem Sinn, daß es keine ω-Relation gibt, die alle möglichen Antworten repräsentiert und deren zuhörige Menge möglicher Relationen echt in der Menge möglicher Relationen der Ergebnisrelation enthalten (und damit informativer) ist.

Diese Eigenschaften sind im Unterschied zu den Eigenschaften, die für Operationen in einem Repräsentantensystem verlangt werden (siehe Abschnitt 4.2), nur für die Charakterisierung einzelner Operationen definiert und im allgemeinen nicht für beliebige Ausdrücke geeignet.

Beispiel 5.1: Seien R = {(ω,3)}, S = {(1,3)} und T = {(2,3)} ω-Relationen über {A,B}. Betrachte folgenden Algebraausdruck:

(R \ S) ∪ (R \ T).

(ω,3) ist gesichertes Antworttupel für diesen Ausdruck. Für die beiden Teilausdrücke R \ S und R \ T darf das Ergebnis aber (ω,3) jeweils nur als mögliches Tupel enthalten, da die leere Relation über {A,B} für beide Ausdrücke eine mögliche Antwort ist. Das Endergebnis enthält deshalb (ω,3) ebenfalls nur als mögliches und nicht als gesichertes Tupel.

Das Beispiel zeigt, daß sich die genannten Eigenschaften im allgemeinen nicht von einzelnen Operationen auf Ausdrücke übertragen. Damit eröffnen sich Möglichkeiten für Operationsdefinitionen, die genauere Ergebnisse im Sinne der Eigenschaften hinreichend und beschränkt liefern als die in [Bis83] angegebenen Definitionen, wenn Operanden durch Ausdrücke gegeben sind.

Eine weitere Einschränkung der in [Bis83] vorgeschlagenen Operationen besteht darin, daß Operanden stets als unabhängig voneinander angesehen werden. Diese Annahme führt dazu, daß etwa eine Definition der Durchschnittsoperation die Eigenschaften hinreichend und beschränkt haben kann, ohne für R ∩ R als Ergebnis die Relation R zu liefern, wenn in R nicht totale Tupel vorkommen. In unseren bisherigen Betrachtungen haben wir eine solche Unabhängigkeit zunächst ebenfalls angenommen, dann aber auch diskutiert, wie identische Tupel bei der Auswertung Berücksichtigung finden können. Wir werden auch in diesem Kapitel in derartiger Weise vorgehen.

Betrachten wir die beiden Eigenschaften hinreichend und beschränkt, wenn die Vervollständigung als POSS-Funktion für Zustände angenommen wird. Für alle ω-Zustände z_ω zu einem beliebigen DB-Schema σ gilt $POSS_c(z_\omega) \subseteq POSS_{ce}(z_\omega)$. Daher sind unter der engen Ausdehnung hinreichende Operationen auch unter der Vervollständigung hinreichend. Für die Beschränktheit von Operationen gilt dies nicht, wie am Beispiel der Differenz deutlich wird. Betrachte hierzu die Differenz {(1,2,s), (1,3,s)} \ {(1,ω,s)}. Mit der engen Ausdehnung als Z-POSS-Funktion ist das

Ergebnis mit einer hinreichenden und beschränkten Definition der Differenz {(1,2,m), (1,3,m)}. Damit ist insbesondere die leere Relation eine mögliche Antwort. Mit der Vervollständigung als Z-POSS-Funktion ist dagegen die leere Relation keine mögliche Antwort.

5.2 Die Unentscheidbarkeit gesicherter Antworten

Sowohl mit der Vervollständigung als auch mit der engen Ausdehnung als POSS-Funktion ist die Menge möglicher Relationen zu einer gegebenen ω-Relation immer dann unendlich, wenn ein unbekannter Wert in einem Attribut vorkommt, dessen Wertebereich unendlich ist. Für eine Anfrage q an ein DB-Schema σ mit einem solchen Wertebereich liegt der Definition der gesicherten Antwort auf q in einem ω-Zustand z_ω zu σ daher eine unendliche Menge von Zuständen zugrunde, wenn in z_ω ein unbekannter Wert vorkommt. Mit der Vervollständigung als Z-POSS-Funktion können trotzdem zu jeder erlaubten TRC-Anfrage gesicherte Antworten vollständig berechnet werden. Dies hängt damit zusammen, daß die Anzahl der möglichen Zustände zwar unendlich sein kann, die Anzahl von Tupeln in jedem derartigen Zustand aber durch die Anzahl der Tupel in der Relation des ω-Zustands beschränkt ist. Wir werden darauf im nächsten Kapitel näher eingehen.

Mit der engen Ausdehnung als Z-POSS-Funktion sind gesicherte Antworten dagegen im Fall unendlicher Wertebereiche im allgemeinen nicht mehr berechenbar, wie wir im folgenden zeigen werden. Hier ist die Anzahl der Tupel in jedem möglichen Zustand zwar ebenfalls endlich, es gibt für sie im allgemeinen aber keine obere Schranke. Dadurch kommt der Satz von Trachtenbrot ([Tra50], siehe hierzu z.B. [EFT92]) zum Tragen.

Wir betrachten im folgenden gesicherte totale Antworten. Da in jeder der von uns betrachteten Formen gesicherter Antworten die totale Antwort enthalten ist, gilt die Unentscheidbarkeitsaussage auch für diese Antwortformen.

Sei $q = (x)/ R\ x \wedge \Phi(x)$ eine TRC-Anfrage an ein DB-Schema σ. Für einen beliebigen ω-Zustand z_ω von σ ist ein totales Tupel $t \in z_\omega(R)$ in der gesicherten totalen Antwort auf q im Zustand z_ω enthalten, wenn $\Phi(t)$ *wahr* ist in jedem möglichen Zustand von z_ω. Zu entscheiden, ob ein solches Tupel t in der gesicherten totalen Antwort enthalten ist, ist daher äquivalent damit zu entscheiden, ob $\Phi(t)$ eine Tautologie ist bezüglich einer gegebenen Zustandsmenge. Eine solche Zustandsmenge kann unendlich sein, wenn einer oder mehrere Wertebereiche von Attributen des DB-Schemas σ unendlich sind. Da Zustände Interpretationen von TRC-Ausdrücken festlegen, haben wir eine Problemstellung, deren Unentscheidbarkeit aus der Logik wohlbekannt ist. Dabei genügt es, erlaubte TRC-Anfragen zu betrachten, in denen jede Variable durch genau einen Bereichsausdruck beschränkt ist.

Satz 5.1: Gesicherte totale Antworten zu erlaubten TRC-Anfragen in ω-Zuständen sind nicht berechenbar, wenn als Z-POSS-Funktion die enge Ausdehnung zugrunde gelegt wird.

Beweis: Der Beweis des Satzes basiert auf dem folgenden Resultat von Trachtenbrot ([Tra50]): "Es ist nicht entscheidbar, ob in der Prädikatenlogik erster Stufe mit oder ohne Gleichheit ein Ausdruck ohne freie Variablen (Satz) in beliebigen endlichen, nichtleeren Wertebereichen gültig ist".

Wir zeigen, daß dieses Entscheidbarkeitsproblem auf das Entscheidbarkeitsproblem für totale gesicherte Antworten zurückgeführt werden kann, wobei wir

der einfacheren Vorgehensweise wegen Anfragen des wertebereichsorientierten Relationenkalküls (DRC) betrachten. Da dieser Kalkül im Fall totaler Zustände die gleiche Ausdruckskraft hat wie der tupelorientierte Kalkül und im Beweis nur Interpretationen für totale Zustände betrachtet werden, gilt der Beweis auch für TRC-Anfragen.

Sei Φ ein beliebiger Satz der Prädikatenlogik erster Stufe. Sei σ ein DB-Schema über einer Attributmenge α mit der Wertebereichsfunktion dom $| \alpha \rightarrow \{\mathbb{N}\}$ derart, daß folgendes gilt: Für jedes Prädikatensymbol P_i von Φ gibt es einen Relationstyp R_i über $\alpha_i \subseteq \alpha$ in σ mit $|\alpha_i| - 1$ = Stelligkeit von P_i; zusätzlich enthält σ einen Relationstyp R_β über einer beliebigen nichtleeren Attributmenge $\beta \subseteq \alpha$ sowie einen Relationstyp R_{DOM} über einem Attribut $A \in \alpha$.

Sei o.B.d.A. die Attributmenge jedes Relationstyps geordnet. Somit können Projektionen in der Form $R_i[\$2,\$3]$ usw. geschrieben werden. Betrachte die folgende DRC-Anfrage:

$$q_\Phi = (x_1,....,x_{|\beta|})/\ R_\beta(x_1,....,x_{|\beta|}) \wedge \Phi_{DRC}$$

wobei Φ_{DRC} aus Φ wie folgt erhalten wird:
Jedes Vorkommen eines atomaren Ausdrucks $P_i(z_1,....,z_{n_i})$ in Φ wird ersetzt durch den Ausdruck $R_i(0,z_1,....,z_{n_i}) \wedge \neg R_i(1,z_1,....,z_{n_i})$;
jedes Vorkommen eines Quantors $(\exists x)$ oder $(\forall x)$ in Φ wird durch den bereichsbeschränkten Quantor $(\exists x: D_U)$ bzw. $(\forall x: D_U)$ ersetzt. D_U ist dabei ein beliebiger DRC-Ausdruck, der für jeden DB-Zustand z zu σ die Menge derjenigen Werte in z bestimmt, die in $z(R_{DOM}) \cup \bigcup_{R_i} \bigcup_{j\in\{2,..,n_i\}} z(R_i)[\$j]$ enthalten sind (dabei sei $z(R)[\$j]$ die Menge aller Werte im j-ten Attribut von z(R)).
Sei z_ω ein ω-Zustand zu σ derart, daß $z_\omega(R_i) = \{(0,\omega,....,\omega),(1,\omega,....,\omega)\}$, $z_\omega(R_{DOM}) = \{(\omega)\}$ und $z_\omega(R_\beta)$ eine beliebige totale Relation über β. Sei ferner eine beliebige nichtleere, endliche Teilmenge D von $\mathbb{N}$ als Wertebereich für Interpretationen von Φ gegeben (es genügt, nur Teilmengen von $\mathbb{N}$ als Wertebereiche zu betrachten). Jede Interpretation I von Φ mit Wertebereich D bestimmt endlich viele mögliche Zustände z_I von z_ω wie folgt:

$$D_U = D \text{ und } I(P_i) = (z_I(R_i)[\$1 = 0])[\$2,....,\$n_i] \setminus (z_I(R_i)[\$1 = 1])[\$2,....,\$n_i].$$

Wegen $D_U = D$ müssen alle Elemente von D, die nicht in einem $z_I(R_i)$ enthalten sind, in $z_i(R_{DOM})$ vorkommen. Dies folgt aus der Festlegung von D_U. Die Differenzbildung und die Unterscheidung aller Tupel im ersten Attribut aller Relationen R_i ist notwendig, um auch leere Prädikate repräsentieren zu können. Direkt mit Relationen kann dies nicht erfolgen, da zu einer ω-Relation nur dann eine leere mögliche Relation gehören kann, wenn die ω-Relation selbst leer ist.

Offensichtlich gilt mit der getroffenen Zuordnung einer Interpretation I zu einer Menge möglicher Zustände von z_ω:

$$\Phi \text{ ist gültig unter I } \Leftrightarrow\ z_I(R_\beta) = q_\Phi(z_I) \quad \text{für alle } z_I.$$

Sei nun ein beliebiger möglicher Zustand z von z_ω gegeben. Dann wird umgekehrt durch z eine Interpretation I_z mit Wertebereich D_z wie folgt eindeutig bestimmt:

$$D_z = D_U \text{ und } I_z(P_i) = (z(R_i)[\$1 = 0])[\$2,....,\$n_i] \setminus (z(R_i)[\$1 = 1])[\$2,....,\$n_i].$$

Wir erhalten: $z(R_\beta) = q_\Phi(z) \Leftrightarrow \Phi$ ist gültig unter I_z.

Zusammenfassend ergibt sich damit:

$\mathfrak{A}_{total}(q_\Phi, z_\omega) = z_\omega(R_\beta) \Leftrightarrow \Phi$ ist gültig in allen endlichen nichtleeren Wertebereichen.

□

Satz 5.1 kann noch dahingehend verschärft werden, daß die Unentscheidbarkeit schon für DB-Schemata mit einem einzigen Relationstyp gilt:

Korollar: Sei ein DB-Schema $\sigma = \{(R, \{A,B,C\})\}$ mit Wertebereichsfunktion dom | $\{A,B,C\} \rightarrow \{\mathbb{N}\}$ gegeben. Es gibt keinen Algorithmus zur Berechnung gesicherter totaler Antworten auf erlaubte TRC-Anfragen in beliebigen ω-Zuständen von σ, wenn als POSS-Funktion die enge Ausdehnung zugrunde gelegt wird.

Beweis: Von L. Kalmar wurde gezeigt (siehe hierzu etwa [BM75]), daß jeder PL1-Ausdruck Φ in einen PL1-Ausdruck Φ' transformiert werden kann, in dem alle Prädikatensymbole die Stelligkeit zwei haben und der äquivalent ist zu Φ bezüglich der Gültigkeit in den gleichen Wertebereichen. Daher genügt es in dem Beweis zu Satz 5.1, nur solche Ausdrücke Φ zu betrachten, in denen alle Prädikatensymbole die Stelligkeit zwei haben (R_{DOM} kann durch einen Relationstyp über einer zweistelligen Attributmenge ersetzt werden). Relationen über der gleichen Anzahl von Attributen mit demselben Wertebereich können aber in einer einzelnen Relation zusammengefaßt werden, indem ein Attribut hinzugenommen wird, dessen Werte die Zugehörigkeit zu einer der ursprünglichen Relationen angeben. □

Bei der Berechnung gesicherter Antworten können wir damit im allgemeinen keine Vollständigkeit erwarten. Es sind also nicht nur Effizienzgründe, die uns zur Beschränkung auf unvollständige Auswertungsverfahren zwingen.

5.3 Berechnung gesicherter Antworten für TRC-Anfragen

5.3.1 Die ω-Auswertung

Die ω-Auswertung, die wir in Kapitel 4 für die Vervollständigung als POSS-Funktion betrachtet haben, nutzt die Information, daß jedes Vorkommen von ω genau einen Wert repräsentiert, nicht aus. Daher kann diese Auswertung ohne Änderung auch mit der engen Ausdehnung als POSS-Funktion übernommen werden, d.h. Satz 4.1 (S. 68) gilt entsprechend.

Satz 5.2: Sei $q = (x)/\ \Phi(x)$ eine erlaubte TRC-Anfrage an ein DB-Schema σ. Dann gilt mit der engen Ausdehnung als POSS-Funktion für jeden ω-Zustand z_ω zu σ:

$\mathfrak{A}_\omega(q, z_\omega) \subseteq \mathcal{GA}_{uni}(q, z_\omega)$.

Beweis: Mit einer Änderung bei der Betrachtung von $\tilde{B}_{\Phi(t'')}$ kann der Beweis zu Satz 4.1 übernommen werden: Es ist zu beachten, daß bei der Bildung von Disjunktionen und Konjunktionen für quantifizierte Teilausdrücke ein Vergleichsausdruck mit einem oder zwei Vorkommen von ω nicht durch einen einzelnen Vergleichsausdruck, sondern im allgemeinen durch eine Disjunktion bzw. eine Konjunktion von Vergleichsausdrücken zu ersetzen ist. Dies ergibt sich daraus, daß mit der engen Ausdehnung als POSS-Funktion ein Vorkommen von ω in einem ω-Zustand durch mehrere definierte Werte in einem möglichen Zustand ersetzt worden sein kann. Die Argumentation an dieser Stelle im Beweis zu Satz 4.1 ändert sich hierdurch aber nicht. □

Die in Kapitel 4 betrachtete Verfeinerung der ω-Auswertung mit Zuordnung eines Booleschen Wertes für Vergleiche, in denen beide Operanden aus dem gleichen Vorkommen von ω im zugrundeliegenden Zustand bestehen, ist allerdings nicht in dieser einfachen Weise möglich. Da bei der engen Ausdehnung Vorkommen von ω Mengen möglicher Werte repräsentieren, muß wie im Beweis zu Satz 5.2 beachtet werden, daß aus einem einzelnen Vergleichsausdruck Disjunktionen bzw. Konjunktionen von Vergleichen entstehen können. Dabei führen Mengen möglicher Werte, die mehr als ein Element enthalten, dazu, daß in diesen Vergleichen auch unterschiedliche Werte miteinander verglichen werden können. Es treten aber in jedem Fall auch Vergleiche auf, in denen ein Wert mit sich selbst verglichen wird. Dieses Wissen kann man sich bei der Auswertung zunutze machen.

Betrachte einen Vergleichsausdruck $x(A) \ominus y(A)$ in einem erlaubten TRC-Ausdruck Φ zu einem DB-Schema σ und einen ω-Zustand z_ω zu σ. Seien x und y durch eine Interpretation I_ω mit dem gleichen ω-Tupel t belegt, für das außerdem $t(A) = \omega$ gelten soll. "Gleich" soll hierbei nicht nur Wertegleichheit bedeuten, sondern auch die gleiche "Herkunft". Das heißt, der Belegung von x und y liegt entweder der gleiche Bereichsausdruck zugrunde oder zumindest für das Attribut A der gleiche definierende Vergleichsausdruck. Da wir für Wertebereiche angenommen haben, daß sie stets genügend viele Elemente enthalten, repräsentiert der Ausdruck $t(A) \ominus t(A)$ Mengen von Vergleichsausdrücken. Für jeden möglichen Zustand ergibt sich die entsprechende Menge aus den Werten, die für das betrachtete Vorkommen von ω bei Bildung der engen Ausdehnung eingesetzt wurde: Jeder Wert dieser Menge wird mit sich selbst und allen anderen Werten verglichen. Betrachten wir nun den Ausdruck $x(A) \ominus y(A)$ in Abhängigkeit davon, ob x und y freie Variablen in Φ sind oder ob eine der Variablen oder beide Variablen quantifiziert sind.

Seien x und y freie Variablen in Φ. Dann ergibt sich in jedem möglichen Zustand z zu z_ω bei der Auswertung von Φ für jede der endlich vielen Vervollständigungen von t, die durch den Zustand und die Anfrage bestimmt sind, eine Belegung von x und y. Dabei können x und y in den durch z festgelegten Interpretationen unabhängig voneinander belegt werden, es gibt aber auch eine Interpretation, durch die x und y mit dem gleichen Tupel belegt werden. Für jeden möglichen Zustand können wir damit eine Interpretation finden, unter der x und y so belegt werden, daß $x.A \ominus y.A$ für $\ominus \in \{\leq, =, \geq\}$ zu *wahr* und für $\ominus \in \{<, \neq, >\}$ zu *falsch* ausgewertet wird. Da jede solche Belegung von x und y eine Vervollständigung von t ist und Belegungen von Variablen unabhängig voneinander sind, kann unter I_ω dem Ausdruck $x(A) \ominus y(A)$ ebenfalls *wahr* bzw. *falsch* als Wert zugewiesen werden.

Sei nun mindestens eine der beiden Variablen quantifiziert, und sei z wieder ein möglicher Zustand zu z_ω. Sei o.B.d.A. x quantifiziert, und sei y freie Variable im Wirkungsbereich des Quantors von x. Bei der Auswertung jeder durch z bestimmten Interpretation, die zu einer Belegung von x und y mit dem gleichen Tupel t paßt, tritt statt $t(A) \ominus t(A)$ eine Disjunktion (falls x Existenz-quantifiziert) bzw. eine Konjunktion (falls x universell quantifiziert) von Vergleichsausdrücken auf. Diese Ausdrücke gehören zu den Überdeckungen von t in z und lassen sich eindeutig dem Ausdruck $x(A) \ominus y(A)$ in Φ zuordnen. Wie oben gilt auch hier, daß in mindestens einem dieser Ausdrücke ein Wert mit sich selbst verglichen wird. Daher ist bekannt, daß in den Disjunktionen bzw. Konjunktionen ein Vergleichsausdruck vorkommt, der in Abhängigkeit vom Vergleichsoperator den Wert *wahr* oder

falsch hat. In einer Disjunktion von Vergleichen gilt, daß die ganze Disjunktion den Wert *wahr* hat, wenn ein Vergleich diesen Wert hat. Ist dagegen nur bekannt, daß ein Vergleich den Wert *falsch* hat, während von allen übrigen kein definierter Wert bekannt ist, dann ist auch der Wert der gesamten Disjunktion *unbekannt*. Analoges gilt für eine Konjunktion.

Wir nutzen dieses Wissen aus, indem wir die dreiwertige Logik um zwei neue Werte erweitern, in denen diese Information "kodiert" ist: *wahr/unbekannt* und *falsch/unbekannt*. Dem Vergleichsausdruck t(A) Θ t(A) wird für $\Theta \in \{\leq, =, \geq\}$ der Wert *wahr/unbekannt* und für $\Theta \in \{<, \neq, >\}$ der Wert *falsch/unbekannt* zugewiesen.

Zur korrekten Ausnutzung der auf diese Weise kodierten Information definieren wir eine fünfwertige Logik mit den Werten der dreiwertigen Logik und den beiden neuen Werten wie folgt:

Seien die fünf Werte der Logik in der nachfolgend angegebenen Reihenfolge geordnet:

falsch, *falsch/unbekannt*, *unbekannt*, *wahr/unbekannt*, *wahr*.

Mit a und b als Bezeichner für jeweils einen dieser Werte definieren wir:

$a \wedge b =_{df} \min(a,b)$

$a \vee b =_{df} \max(a,b)$

$\neg$ *falsch/unbekannt* $=_{df}$ *wahr/unbekannt*

$\neg$ *wahr/unbekannt* $=_{df}$ *falsch/unbekannt*

Die Negation für die übrigen Werte ist wie in der dreiwertigen Logik definiert. Die übrigen Junktoren sind wieder gemäß den in der Booleschen Logik geltenden Äquivalenzen festgelegt.

Sei diese Logik im folgenden als WUFU-Logik bezeichnet. Wir erhalten für sie ein doppeltes Hasse-Diagramm (Abbildung 5.1), das dem Diagramm aus Abbildung 4.1 (S. 76) für die dort verwendete fünfwertige Logik entspricht. Die Definition der Junktoren Konjunktion und Disjunktion über das jeweils minimale bzw. maximale Element gemäß der auf der Menge $\mathfrak{W}$ der Wahrheitswerte vorgegebenen Ordnung bedeutet, daß ($\mathfrak{W}$, $\wedge$, $\vee$) ein Verband ist. Dieser Verband hat *wahr* als größtes und *falsch* als kleinstes Element und ist distributiv. Außerdem gilt $\neg(\neg a) = a$ und $\neg(a \vee b) = \neg a \wedge \neg b$ für alle $a,b \in \mathfrak{W}$.

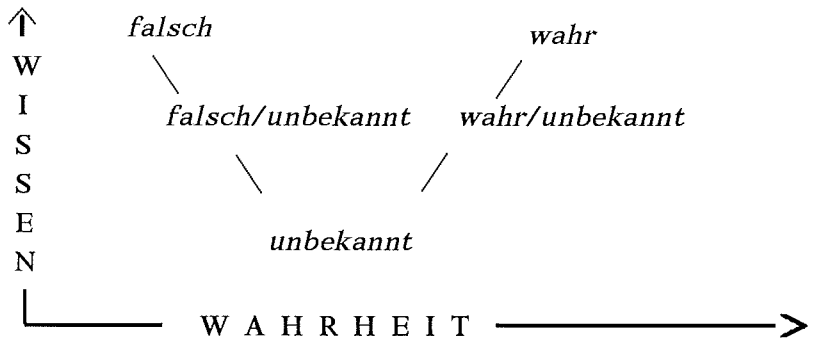


Abb. 5.1: Diagramm für die WUFU-Logik

Bemerkung: Trotz der gemäß der für die Boolesche Logik geltenden Äquivalenz von $a \Rightarrow b$ und $\neg a \vee b$, die nach Festlegung auch für die WUFU-Logik gilt, kann diese Logik nicht als sogenannte *symmetrische Heyting-Algebra* ([BB92]) aufgefaßt werden, da folgende Eigenschaft für die Implikation nicht gilt (wähle z.B. a = *unbekannt* und b = *falsch*):

$a \Rightarrow b$ ist das größte Element in $\mathfrak{W}$, so daß $a \wedge (a \Rightarrow b) \leq b$ gilt.

Damit ist sie auch keine "standard sequence" ([Res69]) und auch keine spezielle Form einer der beiden häufig betrachteten mehrwertigen Kalküle: des n-wertigen Kalküls von J. Lukasiewicz oder desjenigen von E.L. Post (für diese Kalküle siehe [Res69] oder [BB93]). Die WUFU-Logik ist aber identisch mit einer Verallgemeinerung der Kleene'schen dreiwertigen Logik (der von uns betrachteten Variante), die in [Res69] *variant-standard system* genannt und in [Muk81] als ein Modell einer Folge von Modellen von *Fuzzy-Algebren* eingeführt wird.

Für den allgemeinen Fall mit p Werten in $\mathfrak{W}$ können *variant-standard systems* bzw. *p-wertige Kleene-Logiken* wie folgt definiert werden: Sei $\mathfrak{W} = \{0, 1, 2, ..., p-1\}$, und seien $a,b \in \mathfrak{W}$;

$$a \vee b = \max(a,b), \quad a \wedge b = \min(a,b), \quad \neg a = p-1 - a.$$

Die übrigen Junktoren sind wieder über die klassischen Äquivalenzen definiert. Diese Definition entspricht für den Fall p = 5 genau unseren Festlegungen. Damit können z.B. die in [Muk81] angegebenen Äquivalenzen für Fuzzy-Algebren bei der Umformung von Anfragen genutzt werden.

Verallgemeinerungen der Kleene'schen Logik auf mehr als drei Werte werden auch in [Bel77] und [Fit91] diskutiert. Für diese Erweiterungen werden zusätzlich Operatoren bzgl. der zweiten Ordnung, der Wissensordnung, angegeben. P-wertige Kleene-Logiken lassen sich als Unterlogiken dieser Erweiterungen betrachten.

Bemerkung: Wir bleiben trotz dieser Übereinstimmung mit einer bekannten Logik bei der Bezeichnung WUFU-Logik, da wir mit ihr auch eine spezielle Auswertung von Disjunktionen und Konjunktionen verbinden (s.u.).

Eine Auswertung von Anfragen, die sich an der ω-Auswertung orientiert, kann diese Logik wie folgt nutzen:

- Bei der Ersetzung von quantifizierten Teilausdrücken durch Disjunktions- und Konjunktionsausdrücke wird jeder solche Ausdruck geeignet markiert (z.B. mit $\exists$ und $\forall$, siehe Beispiel 5.2 unten).
- Die erzeugten Booleschen Ausdrücke werden unter Zugrundelegung der fünfwertigen Logik ausgewertet, wobei die markierten Ausdrücke von innen nach außen zur Auswertung gelangen.
- Ist der Wert für einen markierten Ausdruck ermittelt, dann wird dieser Wert geändert, falls er *wahr/unbekannt* oder *falsch/unbekannt* ist, und zwar wie folgt:

  Falls es sich um einen Disjunktionsausdruck handelt, wird *wahr/unbekannt* zu *wahr* und *falsch/unbekannt* zu *unbekannt*.

  Falls es sich um einen Konjunktionsausdruck handelt, wird *wahr/unbekannt* zu *unbekannt* und *falsch/unbekannt* zu *falsch*.

Diese Festlegung nutzt genau die Information aus, die durch die neuen Werte repräsentiert wird. Sei die ω-Auswertung so abgeändert, daß die eingeführte fünfwertige Logik in der beschriebenen Weise angewandt wird. Aus den Betrachtungen folgt:

Satz 5.3: Sei q = (x)/ Φ(x) eine erlaubte TRC-Anfrage an ein DB-Schema σ. Die ω-Auswertung liefert unter Verwendung der WUFU-Logik für jeden Zustand z_ω zu σ eine Antwort $\mathfrak{A}_{WUFU}(q,z_\omega)$ auf q mit

$$\mathfrak{A}_\omega(q,z_\omega) \subseteq \mathfrak{A}_{WUFU}(q,z_\omega) \subseteq \mathcal{GA}_{uni}(q,z_\omega).$$

Beispiel 5.2: Sei R = {(ω,3), (2,4)} ein ω-Relation über {A,B}. Betrachte folgende TRC-Anfrage:

(x)/ R x ∧ ¬(∀y)(R y ⇒ (y.A = x.A ⇒ y.B > 3)) oder äquivalent

(x)/ R x ∧ ¬(∀y)(R y ⇒ (y.A ≠ x.A ∨ y.B > 3))

Für die Belegung von x mit (ω,3) erhalten wir:

(ω,3) ∈ R ∧ ¬((ω ≠ ω ∨ 3 > 3) ∧ (ω ≠ 2 ∨ 4 > 3))$_\forall$ ≡

W ∧ ¬((FU ∨ F) ∧ (U ∨ W))$_\forall$ ≡

W ∧ ¬(FU)$_\forall$ ≡

W ∧ ¬F ≡

W

5.3.2 Sub-Auswertungen

Bei der I_{sub}-Auswertung unter Annahme der Vervollständigung als Z-POSS-Funktion konnten wir uns die Unterscheidung von ω-Vorkommen in mehrelementige und einelementige Vorkommen zunutze machen. Mit der engen Ausdehnung als Z-POSS-Funktion ist eine derartige Unterscheidung im allgemeinen nicht möglich. Es stellt sich die Frage, ob die Belegung von Variablen mit ω-Tupeln, wie sie bei der I_{sub}-Auswertung vorgenommen wird, bei der Auswertung so genutzt werden kann, daß über die mit der ω-Auswertung berechenbaren Antworten hinausgehende gesicherte Ergebnisse erhalten werden können. Betrachten wir zunächst ein Beispiel.

Beispiel 5.3 (vgl. Beispiel 4.9 (S. 73) und letzten Absatz in 5.1)**:** Seien R = {(1,2), (1,3)} und S = {(1,ω)} ω-Relationen über der Attributmenge {A,B}. Für die TRC-Anfrage

(x)/ R x ∧ ¬(∃y)(S y ∧ y.B = x.B)

ist die Menge gesicherter Antworttupel leer, da es einen möglichen Zustand mit R = S gibt. Wir erhalten also eine andere gesicherte Antwort als mit der Vervollständigung als Z-POSS-Funktion.
Betrachte folgende Anfrage:

(x)/ R x ∧ ¬(∃y)(S y ∧ y.B = x.B) ∨
S x ∧ x.B = 2 ∧ (∃y)(S y ∧ y.B = 3)

Die gesicherte max-Antwort auf diese Anfrage (mit der engen Ausdehnung als Z-POSS-Funktion) ist {(1,ω)}; die gesicherte uni-Antwort ist leer. Ist in einer möglichen Relation zu S entweder (1,2) oder (1,3) oder keines von beiden Tupeln enthal-

ten, dann sorgt der erste Teil der Anfrage dafür, daß entweder (1,3) oder (1,2) oder beide Tupel in der Antwort enthalten sind. Sind (1,2) und (1,3) in einer möglichen Relation zu S enthalten, dann wird durch den zweiten Teil der Anfrage bewirkt, daß (1,2) in der Antwort enthalten ist.

Da mit der engen Ausdehnung als POSS-Funktion in der Menge der möglichen Zustände zu einem ω-Zustand z_ω alle Zustände vorkommen, die mit der Vervollständigung als POSS-Funktion zu z_ω erhalten werden, ist die Menge gesicherter Antworttupel für diese POSS-Funktion stets in der Menge gesicherter Antworttupel bei Zugrundelegung der Vervollständigung als POSS-Funktion enthalten. Wenn wir Logiken verwenden wollen, wie wir sie in Kapitel 4 eingeführt haben, müssen demnach Einschränkungen vorgenommen werden, d.h. es muß öfter *unbekannt* als Wert angenommen werden. Dies zeigt auch das Beispiel von oben.

Im folgenden wollen wir wie in Kapitel 4 untersuchen, welche Information aus Vergleichsausdrücken mit einem mengenwertigen Vorkommen von ω als Operand bei der Auswertung genutzt werden kann. Da gewöhnliche Vorkommen von ω hier für beliebig viele Werte stehen, müssen andere Fallunterscheidungen gemacht werden. Außerdem muß beachtet werden, daß einem einzelnen Vergleichsausdruck innerhalb eines Ausdrucks bei der Auswertung in einem ω-Zustand z_ω eine Menge von Ausdrücken bei der Auswertung in einem möglichen Zustand zu z_ω entsprechen kann.

Beispiel 5.4: Bei der Auswertung der ersten Anfrage aus Beispiel 5.3 erhalten wir mit Übernahme der Vorgehensweise bei der sub-Auswertung in Kapitel 4:

$$(1,\omega_{\{\}}) \in R \ \wedge \ \neg(\omega_{\{\}} = \omega).$$

Für die Auswertung in dem möglichen Zustand mit S = {(1,2), (1,3)} ergibt sich:

$$(1,\omega_{\{\}}) \in R \ \wedge \ \neg(\omega_{\{\}} = 2 \ \vee \ \omega_{\{\}} = 3).$$

Statt eines einzigen Vergleichsausdrucks erhalten wir wegen der Existenz-Quantifizierung von y in der Anfrage für jede Auswertung in einem möglichen Zustand eine Disjunktion von Vergleichsausdrücken. Die Anzahl der Disjunktionsterme ist gleich der Anzahl der Werte, die für das Vorkommen von ω in dem möglichen Zustand stehen. Die Ersetzung eines Vergleichsausdrucks durch einen speziellen Wahrheitswert (etwa *lokal falsch* oder *lokal wahr* wie in Kapitel 4) unabhängig von der Umgebung, in der der Ausdruck vorkommt, ist demnach nicht möglich: Der Wert eines einzelnen Terms wirkt sich in einer Disjunktion anders aus als in einer Konjunktion. Bei der Auswertung in z_ω wissen wir, daß mindestens einer der Disjunktionsterme für eine Auswertung in einem möglichen Zustand immer den Wert *falsch* hat, gleich welcher der Werte 2 und 3 für $\omega_{\{\}}$ eingesetzt wird. Da die übrigen Werte der Disjunktionen aber nicht bekannt sind, kann auf den Gesamtwert nicht geschlossen werden. Für die Anfrage

$$(x)/\ R\ x \ \wedge \ (\forall y)(S\ y \Rightarrow y.B = x.B)$$

kann dagegen aus dem Wissen, daß mindestens einer der Konjunktionsausdrücke immer den Wert *falsch* hat, geschlossen werden, daß die gesamte Konjunktion den Wert *falsch* hat und die Menge gesicherter Antworttupel damit leer ist.

Nicht nur die Art der Quantifizierung kann eine Rolle spielen, auch die Struktur des Wirkungsbereichs eines Quantors ist für mögliche Rückschlüsse auf Werte von Vergleichsausdrücken von Bedeutung.

Beispiel 5.5: Seien R und S wieder wie in den vorangehenden Beispielen. Betrachte folgende Anfrage:

$$(x)/\ R\ x\ \wedge\ (\forall y)(S\ y\ \Rightarrow\ y.B\ \neq\ x.B)$$

Wenn wir die Umformungsregeln der Prädikatenlogik zugrunde legen, ist diese Anfrage äquivalent zur Anfrage

$$(x)/\ R\ x\ \wedge\ \neg\ (\exists y)(S\ y\ \wedge\ y.B\ =\ x.B)$$

von oben. Für den Vergleichsausdruck $y.B \neq x.B$ gibt es sowohl mögliche Zustände, in denen bei Auswertung Konjunktionen gebildet werden, die den Wert *wahr* haben (wähle z.B. $S = \{(1,4), (1,5)\}$) als auch mögliche Zustände, in denen bei Auswertung Konjunktionen enstehen, die den Wert *falsch* haben (wähle z.B. $S = \{(1,2)\}$). Dem Ausdruck muß daher in einer geeigneten Interpretation der Wert *unbekannt* zugeordnet werden.

Betrachte nun die Anfrage

$$(x)/\ R\ x\ \wedge\ \neg\ (\forall y)(S\ y\ \Rightarrow\ \neg\ (y.B\ \neq\ x.B)).$$

In diesem Fall wissen wir, daß in jedem möglichen Zustand in der entsprechenden für den quantifizierten Teilausdruck erzeugten Konjunktion ein Vergleichsausdruck vorkommt, der den Wert *wahr* hat und somit nach Negation den Wert *falsch*. Jede Konjunktion hat damit immer den Wert *falsch*. Wenn wir wie oben dem Ausdruck $y.B \neq x.B$ den Wert *unbekannt* zuweisen, verschenken wir gesicherte Information bei der Auswertung, was in diesem Fall zu einer leeren gesicherten Antwort statt zu $\{(1,\omega)\}$ führen würde.

Um die in dem Beispiel deutlich werdende Abhängigkeit der Nutzbarkeit von Information aus Vergleichsausdrücken von der Struktur des Wirkungsbereichs eines Quantors und der Art des Quantors geeignet berücksichtigen zu können, bietet sich folgende Vorgehensweise an: Einem Vergleichsausdruck mit einem mengenwertigen Vorkommen von ω wird bei der Interpretation in einem ω-Zustand *lokal wahr* oder *lokal falsch* als Wert zugeordnet in Abhängigkeit davon, ob in jedem möglichen Zustand ein Vergleichsterm für den Ausdruck erzeugt wird, der den Wert *wahr* bzw. *falsch* hat. Diese Zuordnung entspricht genau derjenigen, die wir im vierten Kapitel vorgenommen haben. Alle Booleschen Ausdrücke werden mit der in Kapitel 4 eingeführten fünfwertigen LWLF-Logik ausgewertet. Gehört ein solcher Boolescher Ausdruck zu einem universellen Quantor, dann kann der Wert *lokal falsch* in *falsch* und der Wert *lokal wahr* in *unbekannt* abgeändert werden; im Fall eines Existenz-Quantors ist *lokal wahr* durch *wahr* und *lokal falsch* durch *unbekannt* ersetzbar. In beiden Fällen darf eine Ersetzung allerdings nur vorgenommen werden, wenn die Werte *lokal falsch* und *lokal wahr* nicht abhängig sind von einem $\omega_{\{\}}$-Vorkommen in der Belegung einer Variablen, die global bezüglich des quantifizierten Teilausdrucks ist. Dieses Vorgehen entspricht demjenigen bei der WUFU-Logik von oben bis auf den Unterschied bei der Auswertung der Booleschen Ausdrücke, für die wir hier wegen der speziellen Belegung der freien Variablen die andere Logik benötigen. Wir erhalten:

Satz 5.4: Sei $q = (x)/\ \Phi(x)$ eine erlaubte TRC-Anfrage an ein DB-Schema σ. Die I_{sub}-Auswertung liefert unter Verwendung der LWLF-Logik für jeden ω-Zustand z_ω zu σ eine Antwort $\mathfrak{A}_{LWLF}(q,z_\omega)$ auf q mit

$$\mathfrak{A}_{WUFU}(q,z_\omega)\ \subseteq\ \mathfrak{A}_{LWLF}(q,z_\omega)\ \subseteq\ \mathcal{GT}(q,z_\omega).$$

5.4 Auswertung von Algebraausdrücken

Die im vierten Kapitel für die sm- und die switch-Auswertung gegebenen Definitionen der Algebraoperationen lassen sich für die Auswertung von Algebraausdrücken über ω-Zuständen mit der engen Ausdehnung als Z-POSS-Funktion direkt übernehmen, da ihnen die Interpretation aller Vorkommen von ω gemäß der engen Ausdehnung zugrunde liegt. Die Ergebnisse aus Abschnitt 4.4.1 gelten daher entsprechend. Wie in [Bis83] gezeigt wird, sind diese Operationen hinreichend und beschränkt, wenn die Operationen einzeln betrachtet, Operanden als unabhängig voneinander angesehen und Informationen über (als endlich angenommene) Wertebereiche nicht ausgenutzt werden. Es stellt sich die Frage, ob diese Operationsdefinitionen erweitert werden können, um ähnlich wie im vierten Kapitel bessere gesicherte Ergebnisse für Ausdrücke erhalten zu können. Das Ausnutzen von Informationen über Wertebereiche führt zu Komplexitätsproblemen (pigeon hole principle, [Bis83]); außerdem wollen wir an unserer Annahme festhalten, daß Wertebereiche stets unendlich sind und nicht - wie in [Bis83] - mit der Angabe von Zustandsrelationen auf endliche Teilmengen beschränkt werden.

Die am Ende von Abschnitt 4.4.1 diskutierte Berücksichtigung gleicher Vorkommen von ω aus dem Zustand in Operanden ist mit der engen Ausdehnung als Z-POSS-Funktion nicht in derselben einfachen Weise möglich, da Vorkommen von ω hier Mengen von Werten repräsentieren. Es stellt sich die Frage, wie eine der WUFU-Logik entsprechende Berücksichtigung solcher Vorkommen erfolgen kann.

Wir betrachten im folgenden die s- und m-Versionen der Operationen und beziehen uns dabei auf die in Abschnitt 4.4.1 gegebenen Definitionen.

Für die einstelligen Operationen Projektion und Selektion muß die Identifizierbarkeit von ω-Vorkommen lediglich für die Selektion untersucht werden. Dabei ist eine Identifizierbarkeit nur dann möglich, wenn der Operand durch einen Ausdruck bestimmt ist, in dem sowohl wenigstens einmal der Join als auch die Umbenennung vorkommen. Nur dadurch kann ein Vorkommen von ω aus dem Zustand in zwei verschiedenen Attributen eines Tupels als Wert auftreten (redundante Vergleichsausdrücke seien ausgeschlossen).

Selektion

Durch eine Selektionsoperation kann bewirkt werden, daß für jeden möglichen Zustand mindestens ein Tupel aus dem Zwischenergebnis, auf das die Operation angewandt wird, entfernt wird. Dies bedeutet, daß sich die Semantik von ω-Vorkommen bei Anwendung einer Selektionsoperation ändern kann: Die Werte zweier Vorkommen von ω in einem Tupel müssen nicht mehr in beliebiger Weise miteinander kombinierbar sein oder die Anzahl der durch ω in einem der Attribute repräsentierten Werte kann gleich Null sein kann.

Beispiel 5.6: Sei R = {(1,ω), (1,2)} eine ω-Relation über {A,B}. Betrachte folgenden Algebraausdruck:

$$(R \bowtie R[B \rightarrow C])[B > C]$$

Für jede mögliche Relation zu R wird durch die Selektion mindestens ein Tupel aus dem Ergebnis der Join-Operation entfernt: Alle durch ω repräsentierten Werte werden miteinander kombiniert, alle Tupel mit Kombinationen, in denen der Wert unter B nicht größer als der Wert unter C ist, qualifizieren sich nicht; dies sind insbesondere alle Tupel mit gleichen Werten unter B und C. Ein Tupel (1,ω,ω) im

(gesicherten) Ergebnis der Join-Operation mit vorhergehender Umbenennung repräsentiert daher mit Sicherheit eine andere Menge von Tupeln als ein Tupel (1,ω,ω) im (möglichen) Endergebnis.
Betrachte {(1,1) (1,2)} als mögliche Relation zu R. Dann erhalten wir für den Ausdruck eine leere Antwort.
Mit {(1,3), (1,4), (1,2)} ergibt sich dagegen {(1,4,3), (1,4,2), (1,3,2)}.

Um die Information, daß einige Tupel sich mit Sicherheit nicht qualifizieren, berücksichtigen zu können, ändern wir die betroffenen Vorkommen von ω im Ergebnis dieser Operation in ω⁻ ab. In nachfolgenden Differenz-Operationen kann diese Information zur Verbesserung des Ergebnisses ausgenutzt werden. Durch eine anschließende Projektion geht die Information bzgl. der Kombinierbarkeit allerdings wieder verloren, wenn eines der beiden Attribute, unter denen die ω⁻-Werte vorkommen, nicht in der Attributmenge vorkommt, auf die projiziert wird.

Für den Vergleich ω Θ ω zweier Vorkommen von ω, die dem gleichen Vorkommen im betrachteten Zustand entstammen, legen wir fest:

s-Version der Selektion: $\omega \ \Theta \ \omega = \begin{cases} \textit{wahr}, \text{ falls } \Theta \in \{\leq, =, \geq\} \\ \textit{unbekannt}, \text{ falls } \Theta \in \{<, \neq, >\} \end{cases}$

m-Version der Selektion: $\omega \ \Theta \ \omega = \begin{cases} \textit{wahr}, \text{ falls } \Theta \in \{\leq, =, \geq\} \\ \textit{wahr} \text{ und Kennzeichnung der } \omega\text{-Vorkommen im Ergebnis } (\omega^-), \text{ falls } \Theta \in \{<, \neq, >\} \end{cases}$

Für den Vergleich ω⁻ Θ ω⁻ zweier Vorkommen von ω⁻, die auf das gleiche Vorkommen von ω im betrachteten Zustand zurückgehen, legen wir fest:

s-Version der Selektion: $\omega^- \ \Theta \ \omega^- = \textit{unbekannt}$ für alle Vergleichsoperatoren

m-Version der Selektion: $\omega^- \ \Theta \ \omega^- = \textit{wahr}$ für alle Vergleichsoperatoren

Für die s-Version muß *unbekannt* angenommen werden, da ein Vorkommen von ω⁻ auch den Fall bedeuten kann, daß die Menge der repräsentierten Werte leer ist, d.h. daß das betreffende Tupel für einen möglichen Zustand nichts zum Operanden der Operation beiträgt.

Ist einer der beiden Operanden ω und der andere Operand ω⁻, muß ebenfalls der Fall berücksichtigt werden, daß ω⁻ eine leere Menge von Werten repräsentiert. Damit kann für ω Θ ω⁻ (ω⁻ Θ ω entsprechend) die Festlegung in gleicher Weise wie bei ω⁻ Θ ω⁻ erfolgen.

Vereinigung

Sei t ein Teil eines Tupel aus einem gegebenen Zustand, der als Tupel in beiden Operanden einer Vereinigungsoperation vorkommt. Für die Definition der beiden Versionen der Operation Vereinigung ergibt sich keine Änderung aus diesem Wissen gegenüber der gewöhnlichen Definition, weil Duplikate wegen der engen Ausdehnung als Z-POSS-Funktion keiner besonderen Behandlung bedürfen. Da nach Durchführung einer Vereinigungsoperation keine Identifizierung von ω-Vorkommen mehr möglich ist, sind eventuelle Vorkommen von ω⁻ wieder in ω zu verwandeln.

Differenz

Anders als bei der Vereinigung kann bei der Differenz das Wissen um Vorkommen identischer Tupel in den beiden Operanden genutzt werden. Außerdem ist der

Unterschied von ω- und ω^--Vorkommen in (ansonsten) identischen Tupeln relevant. Für die s-Version spielt nur die Unterscheidung von ω^-- und ω-Vorkommen eine Rolle, da bei dieser Version auch partielle Tupel identifiziert werden, wenn sie "syntaktisch" gleich sind. Sei mit $t'[\omega^-/\omega]$ das Tupel bezeichnet, das aus t' durch Ersetzung der Vorkommen von ω^- durch ω entsteht. Sei ferner die Relation Δ auf Tupel mit Vorkommen von ω^- ausgedehnt, indem solche Vorkommen wie Vorkommen von ω betrachtet werden. Seien R und U ω-Relationen über der gleichen Attributmenge α.

$$R \setminus_s U =_{df} \{t \mid t \in R \wedge (\forall t' \in U)(t' \,\Delta\, t \Rightarrow t' \neq t \wedge ident(t,R,t'[\omega^-/\omega],U))\}$$

$$R \setminus_m U =_{df} R \setminus \{t \mid t \in R \wedge (\exists t' \in U)((t' = t \wedge Def(t) = \alpha) \vee ident(t,R,t',U))\};$$

dabei soll das Prädikat ident(u,R,v,U) für zwei Tupel u, v über der gleichen Attributmenge α genau dann *wahr* sein, wenn u = v und für jedes Attribut A aus α mit u(A) = v(A) = ω gilt: Das Vorkommen von ω in $u_{|A}$ und in $v_{|A}$ geht in R und in U auf dasselbe Vorkommen von ω im Zustand zurück. ω^- muß in der Definition nicht berücksichtigt werden, da es gemäß den Operationsdefinitionen in Operanden, die gesicherte Ergebnisse repräsentieren, nicht vorkommen kann.

In das Ergebnis der s-Version der Differenz werden somit auch diejenigen Tupel aufgenommen, für die im Subtrahenden nur ein bis auf ω^--Vorkommen gleiches Tupel enthalten ist. Zu diesem Tupel im Subtrahenden gehört eine eingeschränkte Menge möglicher Tupel, so daß im entsprechenden Zwischenergebnis für jeden möglichen Zustand mindestens ein mögliches Tupel zu dem Tupel im Minuenden existiert.

Im Ergebnis der m-Version kann ein Tupel aus dem Minuenden auch dann (mit Sicherheit) entfernt werden, wenn ein syntaktisch gleiches Tupel im Subtrahenden existiert und alle sich in den beiden Tupeln entsprechenden Vorkommen von ω auf das jeweils gleiche Vorkommen im Zustand zurückgehen.

Join

Für den Join erhalten wir eine geänderte s-Version, während die m-Version sich nicht ändert, da in ihr schon alle Tupel miteinander verknüpft werden, die möglicherweise verknüpfbar sind.
Seien T und U ω-Relationen über den Attributmengen β bzw. γ mit $\beta,\gamma \subseteq \alpha$;

$$T \bowtie_s U =_{df} T[\beta \cap \gamma]_{total} \bowtie U[\beta \cap \gamma]_{total} \bowtie T \bowtie U \;\cup$$
$$\{t \bowtie u \mid t \in T \wedge u \in U \wedge ident(t_{|\beta \cap \gamma},T,u_{|\beta \cap \gamma},U)\}$$

Die Korrektheit dieser Festlegungen ist einfach nachzuprüfen: Eine Identifizierbarkeit von Tupeln wird nur dann angenommen, wenn sie für den gegebenen Ausdruck direkt aus der Syntax folgt bzw. durch Überprüfung allein der Operanden einer auszuwertenden Operation festgestellt werden kann (siehe hierzu Abschnitt 4.4.1). Damit steht jedes Vorkommen von ω bzw. ω^- in diesen Tupeln für die gleiche (unbekannte) Menge von Werten wie im Zustand. Es sind demnach bei den Operationsdefinitionen als mögliche Operanden alle Relationen zu berücksichtigen, die aus den gegebenen Operanden durch unabhängige Ersetzung der beiden $\omega(\omega^-)$-Vorkommen (sowie davon unabhängig eventuell weiterer Vorkommen von ω bzw. ω^-) durch eine endliche Anzahl von Werten des zugehörigen Wertebereichs entstehen können.

Betrachte als Beispiel die Join-Bildung $(1,\omega) \bowtie (1,\omega)$. Hierzu gehören bei Auswertung in möglichen Zuständen die Join-Bildungen

$\{\{(1,n) \bowtie (1,m) \mid n,m \in M\} \mid M$ endliche Teilmenge von $\mathbb{N}\}$,

wobei jedes Element dieser Menge die Join-Bildungen zu genau einem möglichen Zustand angibt. Damit ist offensichtlich, daß für jeden möglichen Zustand alle Tupel $(1,n)$, $n \in M$, enthalten sind, d.h. $(1,\omega)$ kann nicht nur für die m-Version, sondern auch für die s-Version des Join in das Ergebnis aufgenommen werden. In der gleichen Weise kann bei den anderen Modifikationen der Operationen argumentiert werden. Wir erhalten damit:

Satz 5.5: Sei e ein beliebiger Ausdruck der Relationenalgebra zu einem DB-Schema σ, und sei z_ω ein ω-Zustand zu σ. Dann ist das Ergebnis, das durch die Auswertung von e mit den erweiterten Operationsdefinitionen unter Berücksichtigung identischer Vorkommen von ω und mit der gleichen Versionszuordnung wie bei der switch-Methode erhalten wird, in $GT(e,z_\omega)$ enthalten.

Mit den geänderten Operationsdefinitionen erhalten wir für die s- und m-Versionen der Selektion sowie für die s-Version der Differenz und des Join stets ein Ergebnis, das dasjenige umfaßt, das mit den s- und m- bzw. s-Versionen der Operationen bei Anwendung der ω-Auswertung erhalten wird. Für die m-Version der Differenz gilt dagegen die umgekehrte Inklusion. Alle anderen Definitionen blieben unverändert.

Es stellt sich die Frage, ob und in welcher Weise sich die Vorgehensweise beim TRC mit Anwendung der WUFU-Logik in diesen Definitionen widerspiegelt. Betrachten wir dazu zunächst ein TRC-Beispiel und einen entsprechenden Ausdruck der Relationenalgebra.

Beispiel 5.7: Sei $R = \{(1,2), (2,\omega), (1,3), (3,3)\}$ wieder eine Relation über $\{A,B\}$; betrachte folgende TRC-Anfrage:

$(x)/\ R\ x \ \wedge\ (\exists y)(R\ y \wedge y.A \geq 2 \wedge y.B = x.B)$

Mit der ω-Auswertung erhalten wir $\{(1,3), (3,3)\}$ als Antwort. Mit Anwendung der WUFU-Logik enthält die Antwort zusätzlich das Tupel $(2,\omega)$. Der folgende Algebraausdruck ist für totale Zustände äquivalent zur TRC-Anfrage:

$((R \bowtie R[A \rightarrow A'])[A' \geq 2])[A,B]$

Es sind für alle Operationen die s-Versionen herzunehmen. Das Tupel $(2,\omega)$ liefert einen Beitrag zum Ergebnis des Join, da es mit sich selbst verknüpft wird. Bei der Auswertung des TRC-Ausdrucks entspricht dies der Zuweisung des Wertes *wahr/unbekannt* für die Matrix des quantifizierten Teilausdrucks bei Belegung von x und y mit diesem Tupel.

Beispiel 5.8: Um eine Entsprechung für universell quantifizierte Teilausdrücke zu erkennen, betrachten wir folgende TRC-Anfrage:

$(x)/\ R\ x \ \wedge\ \neg(\forall y)(R\ y \Rightarrow (y.A = x.A \ \wedge\ y.C = x.C \Rightarrow y.B \geq 2))$

oder äquivalent:

$(x)/\ R\ x \ \wedge\ \neg(\forall y)(R\ y \Rightarrow (y.A \neq x.A \ \vee\ y.C \neq x.C \ \vee\ y.B \geq 2)).$

Sei $R = \{(\omega,1,2), (2,4,2), (3,3,5)\}$ eine Relation über $\{A,B,C\}$. Wir erhalten mit der WUFU-Logik als Antwort: $\{(\omega,1,2)\}$.

Betrachten wir eine Algebraanfrage, die im Fall totaler Zustände äquivalent zur

TRC-Anfrage ist. Dazu erinnern wir zunächst an eine Möglichkeit zur Formulierung der Division mit den uns zur Verfügung stehenden Operationen.
Seien R und S Relationen über den Attributmengen β bzw. γ; sei weiterhin $\gamma \subseteq \beta$ und $\beta' = \beta \setminus \gamma$. Dann gilt für totale Relationen:

$$R \div S = R[\beta'] \setminus ((R[\beta'] \bowtie S) \setminus R)[\beta']$$

Unter Verwendung dieser Gleichung läßt sich die TRC-Anfrage wie folgt als Algebraanfrage mit Wahl der Operationsversionen gemäß der switch-Auswertung schreiben (statt Umbenennungen verwenden wir eine geeignete Indizierung; die Selektionsbedingung, die sich aus der Matrix in der TRC-Anfrage ergibt, bezeichnen wir mit ρ):
Sei $\alpha_x = \{A,B,C\}$;

$$R_x \setminus_s ((R_x \bowtie R_y)[\rho]_m[\alpha_x] \setminus_m (((R_x \bowtie R_y)[\rho]_s[\alpha_x] \bowtie R_y) \setminus_s (R_x \bowtie R_y)[\rho]_m)[\alpha_x])$$

Alle Join Operationen entsprechen hier der Bildung des direkten Produkts, daher ist eine Unterscheidung in m- und s-Versionen nicht notwendig.
Betrachte zunächst die innere s-Version der Differenz. Im Subtrahenden erscheint das Tupel $(\omega^-,1,2,\omega^-,1,2)$ bewirkt durch die m-Version der Selektion mit ρ als Bedingung. Durch die anschließende Anwendung der s-Version der Differenz wird daher das Tupel $(\omega,1,2,\omega,1,2)$ als einziges Tupel nicht aus dem Minuenden entfernt. Nach Projektion auf A,B,C ergibt sich damit für die m-Version der Differenz nach Auswertung: $R_x \setminus_m \{(\omega,1,2)\}$. Als Endergebnis erhalten wir damit auch hier $\{(\omega,1,2)\}$. Die Entsprechung zur WUFU-Logik findet sich in dem Vermerken des Fehlens von Möglichkeiten zur Kombination von Werten durch ω^- und der Ausnutzung dieser Möglichkeit bei der Differenzbildung.

6 V-Relationen

In Kapitel 2 haben wir bei der Definition von V-Relationen und V-Zuständen darauf hingewiesen, daß der Informationsgehalt einer ω-Relation durch eine geeignete V-Relation repräsentiert werden kann. Der Vorteil von V-Relationen und V-Zuständen liegt darin, daß mit jeder Variablen die Information über die Stelle verknüpft ist, an der die Variable vorkommt. In [IL84] werden zur Ausnutzung dieser Information sogenannte *C-tables* eingeführt, die wir im folgenden *C-Relationen* nennen wollen, da auch hier keine Duplikate erlaubt sind, wenn alle Attribute berücksichtigt werden. Diese Relationen verfügen über ein zusätzliches Attribut, in dem als Werte Boolesche Ausdrücke über Vergleichsausdrücken mit den Variablen und Konstanten vorkommen. Die gewöhnliche Relationenalgebra wird in [IL84] zu einer Algebra über C-Relationen erweitert, indem zu allen Operationen angegeben wird, wie für jedes Ergebnistupel der Boolesche Ausdruck in dem zusätzlichen Attribut zu bilden ist. Insofern können C-Relationen mit ihren Operationen als Verfeinerungen der Relationen betrachtet werden, in denen zwischen möglichen und sicheren Tupeln unterschieden wird (siehe Kapitel 4): Tupel, deren zugehöriger Boolescher Ausdruck eine Tautologie darstellt, sind sichere Tupel, während alle anderen Tupel als mögliche Tupel zu betrachten sind. Dabei handelt es sich bei einem Ausdruck dann um eine Tautologie, wenn er für jede erlaubte Belegung der in ihm auftretenden Variablen den Wert *wahr* hat.

Bevor wir C-Relationen mit ihren Operationen genauer betrachten, wollen wir zunächst zeigen, wie sich die in Kapitel 4 für die Auswertung von Algebraausdrücken unter Annahme der Vervollständigung als Z-POSS-Funktion vorgestellten Verfahren (sm- und switch-Auswertung) auf V-Relationen übertragen lassen. Das Vorhandensein von Variablen in Zuständen gestattet dabei im allgemeinen eine bessere Annäherung an gesicherte Antworten, als dies bei Verwendung eines einzigen Symbols (ω) möglich ist. Im zweiten Abschnitt gehen wir dann näher auf die in [IL84] entwickelten Vorschläge für Operationsdefinitionen ein. Für diese Vorschläge zeigen wir, wie sie genutzt werden können, um gesicherte Antworten der in Kapitel 3 vorgeschlagenen Formen zu erhalten.

Wegen der Bezugnahme auf [IL84], wo ausschließlich die Relationenalgebra betrachtet wird, beschäftigen wir uns mit der Auswertung von TRC-Anfragen erst im Anschluß an die Relationenalgebra. Wir stellen dabei verschiedene Verfahren zur Auswertung von TRC-Anfragen in V-Zuständen mit unterschiedlicher Komplexität vor.

Die beweistheoretische Sichtweise des relationalen Datenmodells ([NG78], [Rei78], [Rei80], [Rei84], [Rei86]), wird gelegentlich als besser geeignet für die Behandlung unbekannter Attributwerte angesehen als die übliche modelltheoretische Sichtweise (zur Gegenüberstellung beider Sichtweisen siehe [Rei84], [GMN84] oder [KK93]), mit der wir uns bisher ausschließlich beschäftigt haben. Wir zeigen, daß ein von R. Reiter vorgeschlagenes Verfahren ([Rei84]) zu den gleichen Ergebnissen führt wie eine der von uns betrachteten Methoden zur Auswertung von TRC-Anfragen. Zum Schluß machen wir noch einige Bemerkungen zu einem Vorschlag ([Lak89]), mit Hilfe der intuitionistischen Logik eine bestimmte Form der Vollständigkeit bei den Antworten zu erhalten.

6.1 Effiziente Auswertung von Algebraausdrücken

Für ω-Relationen und ω-Zustände haben wir in den Kapiteln 4 und 5 Verfahren für die effiziente Auswertung von Algebraausdrücken angegeben. Der Preis für die Effizienz ist die Unvollständigkeit der Verfahren, d.h. es werden nicht immer alle Tupel einer gesicherten Antwort erhalten. Korrektheit in dem Sinn, daß die Verfahren stets gesicherte Antworten liefern, ist aber garantiert.

V-Relationen bieten im Unterschied zu ω-Relationen für die Auswertung von Ausdrücken die zusätzliche Verfügbarkeit der "Stellen", an denen unbekannte Attributwerte im Zustand vorkommen. Damit ist es möglich, Vorkommen unbekannter Werte, die dem gleichen Vorkommen im Zustand entstammen, bei der Auswertung geeignet zu berücksichtigen. Andererseits entstehen bei der vollständigen Berücksichtigung dieser Information Komplexitätsprobleme ([AGK91]), wie man sich leicht klarmachen kann. Sollen die angepaßten Verfahren effizient sein, müssen sie daher ebenfalls unvollständig bleiben.

Wir betrachten im folgenden ein Verfahren zur Auswertung von Algebraausdrücken, das sich an die switch-Auswertung für ω-Zustände anlehnt. Es sind also für jede Operation der Relationenalgebra zwei Varianten zu definieren: eine s-Variante und eine m-Variante. Bevor wir zu den Definitionen der Operationen kommen, wollen wir mit Beispielen demonstrieren, welche neuen Möglichkeiten sich durch V-Relationen als Operanden ergeben. Diese V-Relationen repräsentieren Ergebnisse von beliebigen Ausdrücken; deshalb erfüllen sie im allgemeinen nicht mehr die Bedingung für Zustandsrelationen, daß keine Variable mehr als einmal in einer Relation vorkommen darf.

Da jede Variable einen "Platzhalter" für genau einen unbekannten Wert darstellt, können Variable wie Konstanten betrachtet werden mit dem Unterschied, daß Vergleiche von Variablen mit anderen Variablen und Konstanten nicht definiert sind. Das heißt, für eine Variable v gilt zwar $v = v \equiv$ *wahr* und $v \neq v \equiv$ *falsch*, für unterschiedliche Variable v, v' gilt aber weder $v = v' \equiv$ *falsch* noch $v \neq v' \equiv$ *wahr*. Aus der Bedeutung von Variablen ergibt sich vielmehr, daß solchen Vergleichen verschiedener Variablen der Wert *unbekannt* zugeordnet werden muß. Entsprechendes gilt für die übrigen Vergleichsoperatoren.

Beispiel 6.1: Sei $T = \{(1,\omega_1,\omega_1), (\omega_1,2,3), (\omega_2,2,\omega_3)\}$ eine V-Relation über der Attributmenge {A,B,C}.
Für den Ausdruck T[A=1] ist das Tupel $(1,\omega_1,\omega_1)$ mit Sicherheit im Ergebnis enthalten, während die beiden anderen Tupel von T möglicherweise darin enthalten sind, d.h es gibt Belegungen, für die eines der Tupel oder beide die Selektionsbedingung erfüllen.
Für T[B=C] ist $(1,\omega_1,\omega_1)$ mit Sicherheit und $(\omega_2,2,\omega_3)$ möglicherweise im Ergebnis enthalten.
Für T[A=B] ist kein Tupel mit Sicherheit im Ergebnis enthalten. Für die beiden ersten Tupel gibt es jeweils Belegungen, so daß die Selektionsbedingung erfüllt wird, allerdings existiert keine Belegung, für die beide Tupel die Bedingung erfüllen. Gesichertes Wissen ist daher, daß höchstens zwei Tupel in jeder möglichen Antwort enthalten sind.

Beispiel 6.2: Seien $R = \{(1,\omega_1,3), (\omega_2,1,\omega_2)\}$ und $S = \{(\omega_1,1,\omega_3)\}$ V-Relationen über der gleichen Attributmenge {A,B,C}.
Betrachten wir den Join beider Relationen, der hier mit dem Schnitt übereinstimmt.

Das erste Tupel von R ist mit dem Tupel von S verbindbar, wenn ω_1 den Wert 1 hat und ω_3 den Wert 3. Damit ist (1,1,3) das einzige mögliche Ergebnis dieser Verknüpfung. Betrachten wir das zweite Tupel von R und das Tupel von S, dann sind die beiden Tupel nur verknüpfbar, wenn $\omega_1 = \omega_3$ gilt. Ein mögliches Ergebnis ist daher $(\omega_2,1,\omega_2)$. Zusätzliche Bedingung ist aber in diesem Fall, daß im Zustand $\omega_1 = \omega_2 = \omega_3$ gilt. Dieses Tupel kann somit nicht zusammen mit (1,1,3) als mögliches Ergebnis auftreten.

Die Beispiele zeigen die Möglichkeiten, die sich aus der Verwendung von Variablen für Schlußfolgerungen bei der Auswertung ergeben. Sie machen aber auch deutlich, daß nicht alle Schlußfolgerungen berücksichtigt werden können, wenn auf Effizienz geachtet werden soll. Bei den folgenden Operationsdefinitionen wird bewußt auf Information in Ergebnissen verzichtet, damit Operationen mit der gleichen Komplexität wie die entsprechenden Operationen der Relationenalgebra auf totalen Relationen erhalten werden. Falls Operationssymbole keine Indizierung tragen, wird die gewöhnliche Operationsdefinition zugrunde gelegt, wobei Variable wie Konstanten behandelt werden.

Seien R und U V-Relationen über der gleichen Attributmenge α.

1) Vereinigung: $R \cup_s U =_{df} R \cup_m U =_{df} R \cup U$

2) Differenz: $R \setminus_s U =_{df} \{t \mid t \in R \wedge (\forall t' \in U)(\neg (t' \Delta t))\}$

oder äquivalent

$R \setminus_s U =_{df} \{t \mid t \in R \wedge (\forall \nu: \nu \text{ Belegung für } \alpha,V)(\nu(t) \not\in \nu(U))\}$

$R \setminus_m U =_{df} \{t \mid t \in R \wedge \neg (\exists t' \in U)(t' = t)\}$

oder äquivalent

$R \setminus_m U =_{df} \{t \mid t \in R \wedge (\exists \nu: \nu \text{ Belegung für } \alpha,V)(\nu(t) \not\in \nu(U))\}$

Beispiele: $\{(3,\omega_1,\omega_2)\} \setminus_s \{(\omega_1,4,\omega_3)\} = \{(3,\omega_1,\omega_2)\}$

$\{(3,\omega_1,\omega_2), (3,\omega_1,\omega_1)\} \setminus_s \{(\omega_3,2,3)\} = \{(3,\omega_1,\omega_1)\}$

$\{(3,\omega_1,\omega_2), (3,\omega_1,\omega_1)\} \setminus_m \{(\omega_3,2,3)\} = \{(3,\omega_1,\omega_2), (3,\omega_1,\omega_1)\}$

3) Projektion: Sei $\beta \subseteq \alpha$; $R[\beta]_s =_{df} R[\beta]_m =_{df} R[\beta]$

4) Selektion: Sei vga ein einfacher Vergleichsausdruck der Form $A \ominus c$, $c \ominus A$ oder $A \ominus B$ mit $A,B \in \alpha$, dom(A) = dom(B) und $c \in$ dom(A).

Ein V-Tupel t über α erfüllt vga mit Sicherheit, wenn folgendes gilt:
- falls vga gleich $A \ominus c$ oder $c \ominus A$ ist: t(A) ist definiert und steht in der angegebenen Vergleichsrelation zu c;
- falls vga gleich $A \ominus B$ ist: t(A) und t(B) sind definiert und stehen in der angegebenen Vergleichsrelation oder $t(A) = t(B) = v$, $v \in V$ und $\ominus \in \{\leq, =, \geq\}$

Ein V-Tupel t über α erfüllt vga möglicherweise, falls folgendes gilt:
- falls vga gleich $A \ominus c$ oder $c \ominus A$ ist: t(A) ist eine Variable;
- falls vga gleich $A \ominus B$ ist: $\ominus$ ist eine beliebige Vergleichsoperation, und entweder t(A) oder t(B) ist eine Variable oder t(A) und t(B) sind unterschiedliche Variable.

Ein V-Tupel t über α erfüllt vga mit Sicherheit nicht, wenn es vga nicht möglicherweise erfüllt.

$R[vga]_s =_{df} \{t \mid t \in R \wedge t \text{ erfüllt vga mit Sicherheit}\}$

$R[vga]_m =_{df} R[vga]_s \cup \{t \mid t \in R \wedge t \text{ erfüllt vga möglicherweise}\}$

5) Join: Seien T und U V-Relationen über β bzw. γ mit $\beta,\gamma \subseteq \alpha$.

$T \bowtie_s U =_{df} T \bowtie U$

$T \bowtie_m U =_{df} \{t \mid t \text{ V-Tupel über } \beta \cup \gamma \wedge (\exists t' \in T)(\exists t'' \in U)(t'_{|\beta\cap\gamma} \Delta t''_{|\beta\cap\gamma} \wedge$
$t_{|\beta\cap\gamma} = merge(t'_{|\beta\cap\gamma}, t''_{|\beta\cap\gamma}) \wedge t_{|\beta\setminus\gamma} = \hat{t}'_{|\beta\setminus\gamma} \wedge t_{|\gamma\setminus\beta} = \hat{t}''_{|\gamma\setminus\beta}\}$;

dabei sind $\hat{t}'_{|\beta\setminus\gamma}$ und $\hat{t}''_{|\gamma\setminus\beta}$ die Tupel, die aus $t'_{|\beta\setminus\gamma}$ bzw. $t''_{|\gamma\setminus\beta}$ erhalten werden, indem alle Variablen, die auch in $t'_{|\beta\cap\gamma}$ oder $t''_{|\beta\cap\gamma}$ vorkommen, in gleicher Weise wie bei der Bildung von $merge(t'_{|\beta\cap\gamma}, t''_{|\beta\cap\gamma})$ ersetzt werden.

Beispiele: Seien $R = \{(3,\omega_1,\omega_2), (2,\omega_1,\omega_1)\}$ und $U = \{(\omega_3,2,3)\}$ V-Relationen über {A,B,C}; dann gilt: $R \bowtie_s U = \emptyset$ und $R \bowtie_m S = \{(3,2,3)\}$.
Seien $R = \{(3,\omega_1,\omega_1)\}$ und $U = \{(\omega_1,4), (3,2)\}$ V-Relationen über {A,B,C} bzw. {C,D}; dann gilt: $R \bowtie_s U = \{(3,\omega_1,\omega_1,4)\}$ und $R \bowtie_m U = \{(3,\omega_1,\omega_1,4), (3,3,3,2)\}$.

Die Definitionen der Operationen sind so gewählt, daß die Komplexität der Operationen die gleiche ist wie für gewöhnliche Operationen. Der einzige Unterschied besteht darin, daß bei der s-Version der Differenz und der m-Version des Join beim Vergleich von Tupeln nicht auf Gleichheit der Werte, sondern auf ihre Verträglichkeit geprüft werden muß und daß bei der Selektion mehr Fälle unterschieden werden müssen. Der Preis für diese Effizienz ist der Verzicht auf Information, die in den über die Variablen gegebenen Abhängigkeiten zwischen Tupeln einer Relation steckt. Betrachten wir hierzu folgendes Beispiel:
Seien $R = \{(3,\omega_1), (2,\omega_1)\}$ und $U = \{(3,4), (2,3)\}$ V-Relationen über {A,B} bzw. {B,C}. Wir erhalten dann $R \bowtie_m U = \{(3,3,4), (3,2,3), (2,3,4), (2,2,3)\}$.
Gesicherte Information ist aber, daß für keine mögliche Relation zu R alle vier Tupel zusammen im Ergebnis auftreten. Als Ergebnisse sind nur {(3,3,4), (3,2,3)} und {(2,3,4), (2,2,3)} möglich.

Durch die Nichtbeachtung von Abhängigkeiten wird nur die Vollständigkeit von Ergebnissen betroffen, nicht ihre Korrektheit, da sie dazu führt, daß in gesicherten Ergebnissen Tupel fehlen und in möglichen Ergebnissen Tupel hinzukommen können. Für die switch-Auswertung von Ausdrücken bedeutet dies, daß Tupel von gesicherten Antworten im Ergebnis eines Ausdrucks fehlen können, das Ergebnis aber keine Tupel enthalten kann, die nicht zur gesicherten Antwort gehören.

Die Definition der m-Version des Join kann durch eine geeignete Wahl der Variablen in den gemeinsamen Attributen beim Verbinden von Tupeln im Hinblick auf gesicherte Antworten noch verbessert werden. Betrachten wir dazu ein Beispiel.
Seien $R = \{(\omega_1)\}$, $U = \{(\omega_2)\}$ und $T = \{(\omega_3)\}$ V-Relationen über {A}.
Betrachte den Ausdruck

$$R \setminus (((U \bowtie (R \bowtie R[A \rightarrow B])) \bowtie (T \bowtie (R \bowtie R[A \rightarrow B]))[A \neq B])[A]$$

Mit Ausnahme der Differenz sind für alle Operationen die m-Versionen zu nehmen. Wir erhalten bei Auswertung des Subtrahenden (auf die Indizierung der Operationssymbole mit m verzichten wir):

$$(((\{(\omega_2)\} \bowtie (\{(\omega_1)\} \bowtie \{(\omega_1)\})) \bowtie (\{(\omega_3)\} \bowtie (\{(\omega_1)\} \bowtie \{(\omega_1)\})))[A \neq B])[A] =$$

$$(((\{(\omega_2)\} \bowtie \{(\omega_1,\omega_1)\}) \bowtie (\{(\omega_3)\} \bowtie \{(\omega_1,\omega_1)\}))[A \neq B])[A] =$$

$$(((\{(\omega_2,\omega_1)\} \bowtie \{(\omega_3,\omega_1)\})[A \neq B])[A] =$$

(*) $$(((\{(\omega_2,\omega_1)\})[A \neq B])[A] =$$

$$(\{(\omega_2,\omega_1)\})[A] =$$

$$\{(\omega_2)\}$$

Für den gesamten Ausdruck erhalten wir damit die leere Relation über {A} als Ergebnis. Nimmt man bei der Join-Bildung stets ω_1 als Variable im Ergebnis, dann ergibt sich statt des mit (*) gekennzeichneten Ausdrucks:

$(((\{(\omega_1,\omega_1)\})[A \neq B])[A]$ und damit weiter:

$(\Phi)[A] =$

Φ

Wir erhalten somit $\{(\omega_1)\}$ als Ergebnis für den gesamten Ausdruck.

Trotz dieses möglichen Informationsverlustes bleiben wir der Einfachheit halber bei der oben angegebenen Definition. Es sei nur darauf hingewiesen, daß, ohne höheren Aufwand zu verursachen, Heuristiken der folgenden Form verwendet werden könnten: Wähle bei Ersetzungen von Variablen durch Variable immer diejenige Variable mit dem kleineren Index.

Sei mit $\mathfrak{A}_{switch}(e,z_V)$ die Menge der V-Tupel bezeichnet, die durch Anwendung der switch-Auswertung mit den angegebenen Operationsdefinitionen für einen Algebraausdruck e zu einem DB-Schema σ im Zustand z_V erhalten wird.

Mit $\mathfrak{A}_m(e,z_V)$ wollen wir die Tupelmenge bezeichnen, die erhalten wird, wenn für die äußerste Operation bei der Auswertung von e die m-Version verwendet und ansonsten wie bei der switch-Auswertung verfahren wird.

Satz 6.1: Sei e ein beliebiger Ausdruck der Relationenalgebra zu einem DB-Schema σ, und sei z_V ein V-Zustand zu σ. Dann gilt

$$\mathfrak{A}_{switch}(e,z_V) \subseteq \mathcal{GA}_{uni}(e,z_V)$$

und

$$(\forall \nu\text{: } \nu \text{ Belegung für } \alpha,V)(\exists M \subseteq \mathfrak{A}_m(e,z_V))(\nu(M) = e(\nu(z_V))).$$

Beweis: Wir zeigen die beiden Behauptungen des Satzes durch Induktion über die Schachtelungstiefe d von e.

Sei d = 0. In diesem Fall besteht der Ausdruck e aus dem Bezeichner eines Relationstyps von σ. Als Ergebnis erhalten wir den Inhalt der Relation im betrachteten Zustand zu σ. Damit sind beide Behauptungen trivialerweise erfüllt.

Sei d > 0 und die Behauptung für alle Ausdrücke mit Schachtelungstiefe $\leq$ d - 1 bewiesen. Wir betrachten nacheinander die einzelnen Operatoren. Dabei zeigen wir für die jeweilige s-Version den ersten Teil der Behauptung des Satzes und für die m-Version den zweiten Teil.

1) Vereinigung: $e_1 \cup e_2$

i) Die s-Version sei zu berechnen. Beide Operanden sind nach Konstruktion des Operatorbaumes zu e gesicherte Ergebnisse. Es ist die gewöhnliche Vereinigung durchzuführen mit Variablen betrachtet wie Konstanten. Aus der Definition gesicherter uni-V-Antworten folgt direkt:

$$\mathcal{GA}_{uni}(e_1,z_V) \cup \mathcal{GA}_{uni}(e_2,z_V) \subseteq \mathcal{GA}_{uni}(e_1 \cup e_2,z_V)$$

Nach Induktionsannahme gilt:

$$\mathfrak{A}_{switch}(e_i,z_V) \subseteq \mathcal{GA}_{uni}(e_i,z_V),\ i = 1,2.$$

Wir erhalten somit:

$$\mathfrak{A}_{switch}(e_1 \cup e_2,z_V) \subseteq \mathcal{GA}_{uni}(e_1 \cup e_2,z_V).$$

ii) Sei die m-Version zu berechnen.
Nach Induktionsannahme gilt:

$$(\forall \nu\colon \nu \text{ Belegung für } \alpha,V)(\exists M_i \subseteq \mathfrak{A}_m(e_i,z_V))(\nu(M_i) = e_i(\nu(z_V))),\ i = 1,2.$$

Für jedes $t \in (e_1 \cup e_2)(\nu(z_V))$, ν beliebig, aber fest, gilt:

$$t \in e_1(\nu(z_V)) \cup e_2(\nu(z_V)).$$

Damit gibt es $M_i \subseteq \mathfrak{A}_m(e_i,z_V)$, $i = 1,2$, so daß ein $t' \in M_1 \cup M_2$ existiert mit $\nu(t') = t$. Da ν beliebig gewählt war, folgt aus der Definition der m-Version der Vereinigung über die gewöhnliche Vereinigung:

$$(\forall \nu\colon \nu \text{ Belegung für } \alpha,V)(\exists M \subseteq \mathfrak{A}_m(e_1 \cup e_2,z_V))(\nu(M) = (e_1 \cup e_2)(\nu(z_V))).$$

2) Differenz: $e_1 \setminus e_2$.

i) Im Fall der s-Version wird $\mathfrak{A}_{switch}(e_1,z_V) \setminus_s \mathfrak{A}_m(e_2,z_V)$ berechnet. Es ist daher zu zeigen:

$$\mathfrak{A}_{switch}(e_1,z_V) \setminus_s \mathfrak{A}_m(e_2,z_V) \subseteq \mathcal{GA}_{uni}(e_1 \setminus e_2,z_V).$$

Zunächst gilt nach Definition:

$$\mathcal{GA}_{uni}(e_1 \setminus e_2,z_V) = \{t \mid (\forall \nu)(\nu(t) \in (e_1 \setminus e_2)(\nu(z_V)))\} =$$

$$\{t \mid (\forall \nu)(\nu(t) \in e_1(\nu(z_V)) \wedge \nu(t) \notin e_2(\nu(z_V)))\}.$$

Nach Induktionsannahme gibt es zu jeder Belegung ν ein $M \subseteq \mathfrak{A}_m(e_2,z_V)$ mit $\nu(M) = e_2(\nu(z_V))$. Damit kann $\nu(t) \notin e_2(\nu(z_V))$ ersetzt werden durch $t \notin \mathfrak{A}_m(e_2,z_V)$, wenn von der Gleichheit zur Teilmengenbeziehung übergegangen wird, und wir erhalten:

$$\{t \mid (\forall \nu)(\nu(t) \in e_1(\nu(z_V)) \wedge \nu(t) \notin e_2(\nu(z_V)))\} \supseteq$$

$$\{t \mid (\forall \nu)(\nu(t) \in e_1(\nu(z_V)) \wedge t \notin \mathfrak{A}_m(e_2,z_V))\} =$$

$$\mathcal{GA}_{uni}(e_1,z_V) \setminus \mathfrak{A}_m(e_2,z_V), \text{ d.h.}$$

$$\mathcal{GA}_{uni}(e_1,z_V) \setminus \mathfrak{A}_m(e_2,z_V) \subseteq \mathcal{GA}_{uni}(e_1 \setminus e_2,z_V).$$

Aus der Definition der s-Version der Differenz ergibt sich:

$$e_1 \setminus_s e_2 =$$

$\{t \in \mathfrak{A}_{switch}(e_1,z_V) \mid \neg(\exists t' \in \mathfrak{A}_m(e_2,z_V))(t' \Delta t)\} =$

$\mathfrak{A}_{switch}(e_1,z_V) \setminus \{t \mid (\exists t' \in \mathfrak{A}_m(e_2,z_V))(t' \Delta t)\}.$

Da $\mathfrak{A}_{switch}(e_1,z_V) \subseteq \mathcal{GA}_{uni}(e_1,z_V)$ nach Induktionsannahme und da

$\mathfrak{A}_m(e_2,z_V) \subseteq \{t \mid (\exists t' \in \mathfrak{A}_m(e_2,z_V))(t' \Delta t)\}$, folgt

$\mathfrak{A}_{switch}(e_1 \setminus e_2,z_V) =$

$\mathfrak{A}_{switch}(e_1,z_V) \setminus_s \mathfrak{A}_m(e_2,z_V) \subseteq$

$\mathcal{GA}_{uni}(e_1,z_V) \setminus \mathfrak{A}_m(e_2,z_V) \subseteq$

$\mathcal{GA}_{uni}(e_1 \setminus e_2,z_V).$

ii) Im Fall der m-Version wird $\mathfrak{A}_m(e_1,z_V) \setminus_m \mathfrak{A}_{switch}(e_2,z_V)$ berechnet.
Es ist zu zeigen:

$(\forall \nu:\ \nu \text{ Belegung für } \alpha,V)(\exists M \subseteq (\mathfrak{A}_m(e_1,z_V) \setminus_m \mathfrak{A}_{switch}(e_2,z_V)))(\nu(M) = (e_1 \setminus e_2)(\nu(z_V))).$

Nach Induktionsannahme gilt:

$(\forall \nu)(\exists M \subseteq \mathfrak{A}_m(e_1,z_V))(\nu(M) = e_1(\nu(z_V))).$

Da ν eine Funktion ist und da nach Definition der gewöhnlichen Differenz für alle ν gilt:

$(e_1 \setminus e_2)(\nu(z_V)) = e_1(\nu(z_V)) \setminus e_2(\nu(z_V)),$

folgt aus der Annahme:

$(\forall \nu)(\exists M \subseteq \mathfrak{A}_m(e_1,z_V))(\nu(M) = e_1(\nu(z_V)) \setminus e_2(\nu(z_V))).$

Nicht alle Elemente von $\mathfrak{A}_m(e_1,z_V)$ sind notwendig, um geeignete Mengen M zu finden:

$(\forall \nu)(\exists M \subseteq \{t \in \mathfrak{A}_m(e_1,z_V) \mid \nu(t) \notin e_2(\nu(z_V))\})(\nu(M) = (e_1 \setminus e_2)(\nu(z_V))).$

Da $\nu(t) \notin e_2(\nu(z_V))$ für jedes ν gilt, kann die Mengendefinition im Ausdruck wie folgt abgeändert werden ($\{t \in \mathfrak{A}_m(e_1,z_V) \mid ....\}$ vergrößert sich höchstens):

$(\forall \nu)(\exists M \subseteq \{t \in \mathfrak{A}_m(e_1,z_V) \mid (\exists \nu')(\nu'(t) \notin e_2(\nu'(z_V))\})(\nu(M) = (e_1 \setminus e_2)(\nu(z_V))).$

Da $(\forall \nu')(\nu'(\mathcal{GA}_{uni}(e_2,z_V))) \subseteq e_2(\nu'(z_V)))$, wird hierdurch impliziert:

$(\forall \nu)(\exists M \subseteq \{t \in \mathfrak{A}_m(e_1,z_V) \mid (\exists \nu')(\nu'(t) \notin \nu'(\mathcal{GA}_{uni}(e_2,z_V)))\})(\nu(M) = (e_1 \setminus e_2)(\nu(z_V))).$

Nach Induktionsannahme gilt $\mathfrak{A}_{switch}(e_2,z_V) \subseteq \mathcal{GA}_{uni}(e_2,z_V)$, und es folgt:

$(\forall \nu)(\exists M \subseteq \{t \in \mathfrak{A}_m(e_1,z_V) \mid (\exists \nu')(\nu'(t) \notin \nu'(\mathfrak{A}_{switch}(e_2,z_V)))\})(\nu(M) = (e_1 \setminus e_2)(\nu(z_V))).$

Diese Formel ist nach Definition der m-Version der Differenz äquivalent zu

$(\forall \nu)(\exists M \subseteq \mathfrak{A}_m(e_1,z_V) \setminus_m \mathfrak{A}_{switch}(e_2,z_V))(\nu(M) = (e_1 \setminus e_2)(\nu(z_V))),$

was zu beweisen war.

3) Projektion: $e_1[\beta]$.

Die s- und die m-Version der Projektion sind gleich definiert als gewöhnliche Projektion mit Variablen betrachtet wie Konstanten. Für gesicherte uni-V-Antworten gilt offensichtlich $(\mathcal{GA}_{uni}(e_1,z_V))[\beta] \subseteq \mathcal{GA}_{uni}(e_1[\beta],z_V)$, da gesicherte Tupel durch Projektion nicht zu "ungesicherten" Tupeln werden können; es kommen

also höchstens Tupel hinzu. Tupel können hinzukommen, wenn es mehrere Tupel gibt, die auf β', $\beta \subseteq \beta' \subseteq \alpha$, übereinstimmen und für e_1 mögliche, aber nicht gesicherte Tupel darstellen. Damit folgt

$$\mathfrak{A}_{switch}(e_1[\beta],z_V) \subseteq \mathcal{GA}_{uni}(e_1[\beta],z_V)$$

unmittelbar aus der Induktionsannahme.
Da bei der Projektion höchstens Duplikate im Ergebnis entfernt werden, d.h. jedes Tupel des Operanden im Ergebnis "vertreten" ist, folgt auch

$$(\forall \nu)(\exists M \subseteq \mathfrak{A}_m(e_1[\beta],z_V))(\nu(M) = e_1[\beta](\nu(z_V)))$$

aus der Induktionsannahme.

4) Selektion: zu betrachten sind $e_1[A \ominus c]$ und $e_1[A \ominus B]$.

i) Untersuchen wir zunächst die s-Version für beide Formen von Vergleichsausdrücken. Im Fall $A \ominus c$ werden nur Tupel in das Ergebnis übernommen, die in A einen definierten Wert haben, der in der angegebenen Vergleichsrelation zu c steht. Mit der Induktionsannahme folgt damit, daß alle Tupel im Ergebnis in $\mathcal{GA}_{uni}(e_1[A \ominus c],z_V)$ enthalten sind.

Im Fall $A \ominus B$ sind alle Tupel der switch-Auswertung von e_1, deren Werte in A und B definiert sind und in der angegebenen Vergleichsrelation stehen, im Ergebnis enthalten. Diese Tupel sind mit der Induktionsannahme auch in $\mathcal{GA}_{uni}(e_1[A \ominus B],z_V)$ enthalten. Falls $\ominus \in \{\leq, =, \geq\}$, kommen noch solche Tupel hinzu, die in A und B die gleiche Variable als Wert haben. Für jedes ν haben diese Tupel in A und B den gleichen Wert, weshalb auch hier mit der Induktionsannahme folgt, daß sie in $\mathcal{GA}_{uni}(e_1[A \ominus B],z_V)$ enthalten sind.

Andere Tupel werden aus $\mathfrak{A}_{switch}(e_1,z_V)$ in beiden Fällen nicht übernommen.

ii) Betrachten wir nun die m-Version der Operation. Es darf bei Durchführung der Operation kein Tupel verloren gehen, das in einer möglichen Antwort auf $e_1[A \ominus c]$ bzw. $e_1[A \ominus B]$ in z_V enthalten ist. Mit der Induktionsannahme folgt dann der entsprechende Teil des Satzes.

Im Fall $A \ominus c$ werden nur solche Tupel nicht in das Ergebnis übernommen, die in A einen definierten Wert haben, der nicht in der angegebenen Vergleichsoperation zu c steht. Diese Tupel können aber auch in keiner möglichen Antwort auftreten.

Im Fall $A \ominus B$ fehlen nur diejenigen Tupel, deren Tupel in A und B definierte Werte haben, die nicht in der angegebenen Vergleichsrelation stehen, sowie im Fall $\ominus \in \{<, \neq, >\}$ diejenigen Tupel, die in A und B die gleiche Variable als Wert haben. Für jede Belegung ν können diese Tupel nicht in $e_1[A \ominus B](\nu(z_V))$ enthalten sein.

5) Join: $e_1 \bowtie e_2$.
i) Die s-Version des Join verbindet diejenigen Tupel aus den für e_1 und e_2 erhaltenen Ergebnissen, die in den gemeinsamen Attributen identische Konstanten oder identische Variable haben. Nur solche Tupel aus den Ergebnissen für e_1 und e_2 sind auch für jede Belegung ν ihrer Variablen in $e_1(\nu(z_V)) \bowtie e_2(\nu(z_V))$ enthalten und damit Element von $\mathcal{GA}_{uni}(e_1 \bowtie e_2,z_V)$. Mit der Induktionsannahme folgt damit $\mathfrak{A}_{switch}(e_1 \bowtie e_2,z_V) \subseteq \mathcal{GA}_{uni}(e_1 \bowtie e_2,z_V)$.
ii) Durch die m-Version des Join werden aus den für e_1 und e_2 erhaltenen Relationen R_1 und R_2 alle Tupel miteinander verbunden, deren Werte in den gemeinsamen

Attributen jeweils verträglich miteinander sind. Es ist zu zeigen, daß durch die Vorgehensweise keine Tupel verloren gehen können, die für eine beliebige Belegung ν in das Ergebnis aufgenommen werden.

Seien t und t' zwei Tupel aus R_1 bzw. R_2, die bei Anwendung der m-Version des Join für die Auswertung von $R_1 \bowtie R_2$ nicht miteinander verbunden werden. Dann sind t und t' nicht verträglich miteinander auf der gemeinsamen Attributmenge δ, d.h. es gibt keine Belegung ν derart, daß $\nu(t(A)) = \nu(t'(A))$ für alle $A \in \delta$. Damit trägt aber auch $\nu(t) \bowtie \nu(t')$ für kein ν zum Ergebnis von $\nu(R_1) \bowtie \nu(R_2)$ bei.

Falls in einem Attribut der gemeinsamen Attributmenge beide Tupel eine Variable als Wert haben, führt die Übernahme der Variablen aus dem Tupel des ersten Operanden zu keinem Verlust, da für jede Belegung ν das erzeugte Tupel auf den Attributen des ersten Operanden mit dem Tupel aus diesem Operanden übereinstimmen muß. Entsprechendes gilt, wenn in einem gemeinsamen Attribut der eine Operand eine Konstante und der andere Operand eine Variable als Wert hat. In diesem Fall wird beim Verbinden die Konstante als Attributwert eingesetzt. □

Betrachten wir nur die s-Versionen der monotonen Operationen Vereinigung, Projektion, Join und Selektion eingeschränkt auf den Gleichheitsoperator, dann entsprechen die Operationsdefinitionen genau den Definitionen der gewöhnlichen monotonen Algebraoperationen, wenn Variablen wie Konstanten betrachtet werden. In [IL84] wird gezeigt, daß V-Relationen mit den monotonen Operationen ein Repräsentantensystem darstellen. Dies bedeutet, daß die Information, die mit diesen Operationen aus einer Datenbank gewonnen werden kann, reduziert auf den gesicherten totalen Anteil, durch die angegebenen Operationsdefinitionen im Ergebnis von Ausdrücken erhalten bleibt. Wir zeigen im folgenden, daß die switch-Auswertung mit den oben angegebenen Operationsdefinitionen stets Antworten liefert, die der jeweiligen gesicherten uni-V-Antwort entsprechen. Das Ergebnis von [IL84] ergibt sich damit als Korollar.

Seien im folgenden Algebraausdrücke, in denen nur monotone Operationen der oben aufgezählten Art auftreten, als *monotone Ausdrücke* bezeichnet.

In monotonen Operationen werden an zu selektierende oder verknüpfende Tupel bei den Operationen Selektion und Join nur positive Bedingungen an die Gleichheit von Werten gestellt. Daher können bei der Duplikatelimination, die für die Vereinigung und die Projektion notwendig sein kann, nur Disjunktionen solcher Bedingungen an (mögliche oder gesicherte) Tupel impliziert werden. Da Ausdrücke und Relationen endlich sind, ergibt sich aus der Form der möglichen Gleichheitsbedingungen und den Operationsdefinitionen unmittelbar (vergleiche auch Beweis zu Lemma 7.2 in [IL84]):

Lemma 6.1: Sei e^+ ein monotoner Algebraausdruck zu einem DB-Schema σ über einer Attributmenge α, und sei z_V ein V-Zustand zu σ. Dann gilt für jedes Tupel $t \in \mathfrak{A}_m(e^+, z_V)$: Die Menge N_t aller Belegungen ν für α,V mit $\nu(t) \in e^+(\nu(t))$ ist durch einen endlichen Ausdruck der Form

$$\bigvee(\bigwedge \omega_i = c_j \wedge \bigwedge \omega_k = \omega_\ell)$$

bestimmt. Dabei sind die ω_i, ω_j, ω_k Variable aus z_V und die c_j Konstanten aus e^+ oder z_V.

Offensichtlich kann die Menge N aller Belegungen für α,V nicht durch eine endliche Vereinigung von Mengen N_t erhalten werden, wenn die zu den Tupeln t

gehörenden Disjunktionen keine Tautologien darstellen.

Für alle $t \in \mathfrak{A}_m(e^+,z_V) \setminus \mathcal{GA}_{uni}(e^+,z_V)$ ist $N_t \neq N$. Daher gilt mit den eingeführten Bezeichnungen:

Lemma 6.2: $N \neq \bigcup \{N_t \mid t \in \mathfrak{A}_m(e^+,z_V) \setminus \mathcal{GA}_{uni}(e^+,z_V)\}$

Nach diesen Vorbereitungen können wir zeigen, daß im Fall von monotonen Ausdrücken die switch-Auswertung stets die gesicherte uni-V-Antwort liefert. Eine entsprechende Aussage findet sich ohne Beweis schon in [Lip84].

Satz 6.2: Sei e^+ ein monotoner Algebraausdruck an ein DB-Schema σ, und sei z_V ein V-Zustand zu σ. Dann gilt $\mathfrak{A}_{switch}(e^+,z_V) = \mathcal{GA}_{uni}(e^+,z_V)$.

Beweis: Falls alle Operationen monoton sind, werden für die switch-Auswertung nur die s-Versionen der Operationen benötigt. Wir können uns daher bei der folgenden Induktion über die Schachtelungstiefe d von e^+ auf diese Versionen beschränken.

Sei d = 0: trivial.

Sei d ≥ 0 und die Behauptung für alle Ausdrücke mit Schachtelungstiefe ≤ d - 1 bewiesen. Wir betrachten wieder nacheinander die einzelnen Operationen.

1) Vereinigung: $e_1 \cup_s e_2 = e_1 \cup e_2$.

Es ist zu zeigen:

$$\mathcal{GA}_{uni}(e_1 \cup e_2,z_V) = \mathfrak{A}_{switch}(e_1,z_V) \cup \mathfrak{A}_{switch}(e_2,z_V).$$

Es gilt allgemein:

$$\mathcal{GA}_{uni}((e_1 \cup e_2),z_V) =$$
$$\mathcal{GA}_{uni}(e_1,z_V) \cup \mathcal{GA}_{uni}(e_2,z_V) \cup$$
$$\{t \in (\mathfrak{A}_m(e_1,z_V) \setminus \mathcal{GA}_{uni}(e_1,z_V)) \cup (\mathfrak{A}_m(e_2,z_V) \setminus \mathcal{GA}_{uni}(e_2,z_V)) \mid$$
$$(\forall \nu)((\nu(t) \notin e_1(\nu(z_V)) \Rightarrow (\exists t' \in \mathfrak{A}_m(e_2,z_V))(\nu(t') = \nu(t) \wedge \nu(t') \in e_2(\nu(z_V)))) \wedge$$
$$(\nu(t) \notin e_2(\nu(z_V)) \Rightarrow (\exists t' \in \mathfrak{A}_m(e_1,z_V))(\nu(t') = \nu(t) \wedge \nu(t') \in e_1(\nu(z_V))))\}.$$

Begründung: Ein Tupel t aus der Vereinigung der möglichen Antworten zu e_1 und e_2 (ohne jeweils die gesicherten Tupel) kann zur Vereinigung der gesicherten Antworten zu e_1 und e_2 hinzukommen, wenn es in beiden möglichen Antworten Tupel gibt, mit denen t verträglich ist und für die gilt: Falls t für ein ν nicht in der entsprechenden möglichen Antwort enthalten ist, stimmt eines dieser Tupel mit t unter ν überein und ist in der möglichen Antwort für einen der Operanden enthalten. Damit gilt für jedes ν, daß $\nu(t)$ in $(e_1 \cup e_2)(\nu(z_V))$ enthalten ist. Andere Tupel als diejenigen aus der Vereinigung der möglichen Tupel zu e_1 und e_2 können trivialerweise nicht zur gesicherten Antwort beitragen. Statt der Menge der möglichen Tupel kann nach Satz 6.1 das mögliche Ergebnis der switch-Auswertung genommen werden. Aus Lemma 6.2 folgt, daß bei monotonen Ausdrücken diese Menge stets leer ist. Da nach Induktionsannahme

$$\mathfrak{A}_{switch}(e_i,z_V) = \mathcal{GA}_{uni}(e_i,z_V), \; i = 1,2,$$

gilt somit

$$\mathcal{GA}_{uni}(e_1 \cup e_2,z_V) = \mathfrak{A}_{switch}(e_1,z_V) \cup \mathfrak{A}_{switch}(e_2,z_V).$$

2) Projektion: $e_1[\beta]_s = e_1[\beta]$.

Hier kann die Argumentation analog zur Argumentation bei der Vereinigung erfolgen, da durch die Projektionsbildung nur solche Tupel neu zu gesicherten Tupeln werden können, die vor der Projektion schon im Operanden vorkommen und zu denen Duplikate und andere verträgliche Tupel durch die Operation erzeugt werden.

3) Selektion: $e_1[A = c]_s$ oder $e_1[A = B]_s$.

Bei dieser Operation werden für die Bildung der gesicherten uni-V-Antwort eventuell Tupel aus der gesicherten uni-V-Antwort des Operanden entfernt; es kommen nie neue Tupel hinzu. Ein Tupel wird entfernt, wenn seine Werte die angegebene Bedingung nicht erfüllen. In diesem Fall wird es bei der s-Variante der Selektion ebenfalls aus dem Ergebnis entfernt. Mit der Induktionsannahme folgt daher, daß dieses Ergebnis mit der gesicherten uni-V-Antwort übereinstimmt.

4) Join: $e_1 \bowtie_s e_2 = e_1 \bowtie e_2$.

Wiederum können in der gesicherten uni-V-Antwort nur Tupel auftreten, die durch Verknüpfung von Tupeln aus den gesicherten Antworten der Operanden entstehen. Wie bei der Vereinigung und der Projektion können für beliebige Ausdrücke in den Operanden mögliche Tupel vorkommen, die verträglich sind und bei denen sich die Belegungen, für die sie sich qualifizieren, zur Gesamtmenge der Belegungen ergänzen. Da diese Möglichkeit aber für monotone Ausdrücke nicht besteht, können nur Tupel in die gesicherte uni-V-Antwort gelangen, die in den Operanden zur gesicherten Antwort gehören und in der gemeinsamen Attributmenge zu e_1 und e_2 für alle Belegungen übereinstimmen. Diese Übereinstimmung gilt nur dann für alle Belegungen, wenn die Attributwerte identisch sind. Genau in diesem Fall werden verknüpfbare Tupel in das Ergebnis der s-Version des Join übernommen. □

In der gesicherten uni-V-Antwort eines Ausdrucks ist die gesicherte totale Antwort stets enthalten. Wir erhalten demnach ein Ergebnis als Korollar, aus dem die oben genannte Aussage von [IL84] unmittelbar folgt:

Korollar: Sei e^+ ein monotoner Ausdruck der Relationenalgebra zu einem DB-Schema σ, und sei z_V ein V-Zustand zu σ. Dann gilt:

$$\mathcal{GA}_{total}(e^+, z_V) \subseteq \mathfrak{A}_{switch}(e^+, z_V).$$

Jede der angebenen Operationen verursacht die gleichen Kosten wie die Operationen der gewöhnlichen Relationenalgebra. Damit sind totale gesicherte Antworten in polynomieller Zeit berechenbar (in [IL84] als Hauptresultat genannt). Satz 6.2 zeigt, daß dieses Komplexitätsresultat auch für gesicherte uni-V-Antworten gilt.

Satz 6.3: Gesicherte uni-V-Antworten sind für monotone Algebraausdrücke in polynomieller Zeit berechenbar.

Bei der Auswertung von Anfragen in ω-Zuständen haben wir sowohl für die Vervollständigung als auch für die enge Ausdehnung als Z-POSS-Funktion Vorkommen von ω, für die bekannt ist, daß sie mehr als einen Wert repräsentieren, durch das Symbol $\omega_{\{\}}$ gekennzeichnet und bei der Auswertung geeignet berücksich-

tigt. Ein entsprechendes Vorgehen ist auch bei V-Zuständen möglich. Wir wollen darauf aber nicht weiter eingehen, da sich die Vorgehensweisen stark ähneln.

6.2 C-Relationen

Für monotone Ausdrücke haben wir gesehen, daß zu jedem nicht gesicherten Tupel eine Disjunktion von Konjunktionen über Vergleichsausdrücken gehört, die angibt, unter welchen Belegungen das Tupel in der möglichen Antwort enthalten ist. Für beliebige Algebraausdrücke lassen sich derartige Bedingungen für Belegungen ebenfalls angegeben, sie sind lediglich von allgemeinerer Form.

In [IL84] wurden sogenannte *C-tables* eingeführt, die V-Relationen mit einem zusätzlichen Attribut darstellen. In diesem Attribut können die Bedingungen für die einzelnen Tupel als Werte eingetragen werden. Es wurde für die Operationen der Relationenalgebra angegeben, wie sich die Werte in diesem zusätzlichen Attribut bestimmen und wie Duplikate zu behandeln sind. Wir definieren im folgenden zunächst C-tables, die wir C-Relationen nennen wollen, und C-Zustände und geben anschließend die Definitionen aus [IL84] für die Operationen auf C-Relationen an (mit einer Korrektur bei der Differenz). Dabei folgen wir in der Darstellung teilweise [KK93].

Definition: Seien V eine abzählbar unendliche Menge von Variablen und α eine Attributmenge mit Bereichsfunktion dom $| \alpha \rightarrow \{D_1,....,D_k\}$. Seien $V \cap \bigcup_{j=1}^{k} D_j = \phi$ und COND $\notin \alpha$ ein spezielles Attribut mit Wertebereich C_V, wobei C_V die Menge aller Booleschen Ausdrücke über Vergleichsausdrücken der Form $x \ominus y$ ist mit

$$x,y \in V \cup \bigcup_{j=1}^{k} D_j \text{ und } \ominus \in \{ <, \leq, =, \neq, \geq, > \};$$

zusätzlich soll C_V die beiden Wahrheitswerte *wahr* und *falsch* enthalten.

Ein C-Tupel über α ist eine Abbildung

$$t|\alpha \cup \{COND\} \rightarrow \bigcup_{j=1}^{k} D_j \cup V \cup C_V \text{ mit } t(A) \in dom(A) \cup V \text{ für } A \in \alpha \text{ und } t(COND) \in C_V.$$

Eine C-Relation über α ist eine endliche Menge von C-Tupeln über α.

Definition: Ein C-Zustand zu einem DB-Schema $\sigma = \{(RT_1,\alpha_1),....,(RT_m,\alpha_m)\}$ ist eine Abbildung $z_C \mid \sigma \rightarrow \{R_1,....,R_m\}$ mit $z_C((RT_i,\alpha_i)) = R_i$, R_i ist eine C-Relation über α_i, i = 1,....,m, und folgenden Eigenschaften: Eine Variable $v \in V$ tritt höchstens einmal in den Relationen von z_C auf, und für jedes $t \in R_i$, i = 1,....,m, gilt: t(COND) hat den Wert *wahr*.

Offensichtlich kann ein C-Zustand aus einem V-Zustand erhalten werden, indem jeder V-Relation das Attribut COND mit dem Wert *wahr* (oder einer Gleichung der Form c = c, c beliebige Konstante aus einem Wertebereich) für jedes Tupel hinzugefügt wird. In Beispielen lassen wir den Wert für COND weg, falls es sich um Relationen eines C-Zustands handelt.

Um Eindeutigkeit zu gewährleisten, schreiben wir Werte für COND gelegentlich in der Form #....#.

Belegungen lassen sich für C-Relationen in der gleichen Weise wie für V-Relationen definieren. Bei ihrer Anwendung auf eine C-Relation werden auch alle Vorkommen von Variablen im Attribut COND durch die entsprechenden Konstanten ersetzt. Da die Ausdrücke im Attribut COND die Belegungen angeben, unter denen die zugehörigen Tupel in den entsprechenden möglichen Relationen enthalten sind, ergeben sich mögliche Relationen zu C-Relationen wie folgt:

Definition: Seien R eine C-Relation über α und V die Variablenmenge zu α.
Eine totale DB-Relation T über α heißt mögliche Relation zu R, falls es eine Belegung ν für α und V gibt derart, daß

$$T = \{t_{|\alpha} \mid t \in \nu(R) \wedge t(COND) \text{ ist äquivalent zu } \textit{wahr}\}$$

In [IL84] werden einige Regeln für das Zusammenfassen, Streichen und Umformen von Tupeln in C-Relationen angegeben, durch deren Anwendung sich die Menge der jeweils zugehörigen möglichen Relationen nicht ändert. Diese Regeln genügen für die in [IL84] angestrebten Aussagen. Wir werden später noch Regeln hinzufügen, um bestimmte Antwortformen erhalten zu können. Betrachten wir zunächst die gegebenen Regeln.

Umformungsregeln für C-Relationen

1) Treten in einer C-Relation Tupel auf, die sich nur im Attribut COND unterscheiden, dann können diese Tupel zu einem Tupel zusammengefaßt werden mit den gegebenen Werten in den gewöhnlichen Attributen und der Disjunktion aller Ausdrücke im Attribut COND.
2) Alle Tupel, deren Wert in COND einen Widerspruch darstellt (etwa: $c \neq c$), können entfernt werden.
3) Für jedes Tupel darf der Ausdruck in COND durch einen beliebigen äquivalenten Ausdruck ersetzt werden. Zwei Ausdrücke sind äquivalent, wenn sie für alle Belegungen den gleichen Wert haben.

C-Relationen sollen im folgenden stets in der Weise *normiert* sein, daß weder Regel 1 noch Regel 2 anwendbar sind. Von der Regel 3 wird freier Gebrauch gemacht.

Beispiel 6.3: Betrachte folgende C-Relation:

A	B	COND
3	4	$\omega_1 = 3$
3	4	$\omega_1 \neq 3$
5	6	$\omega_1 \neq \omega_1$

Durch Anwendung der Regeln 2) und 3) erhalten wir

A	B	COND
3	4	$\omega_1 = 3 \vee \omega_1 \neq 3$
5	6	*falsch*

und mit den Regeln 1) und 2) dann

A	B	COND
3	4	*wahr*

Nach diesen Vorbereitungen können wir die Definitionen der Operationen der Relationenalgebra für C-Relationen angeben.

Seien P und Q C-Relationen über der Attributmenge α, und sei S eine C-Relation über der Attributmenge β.

1) Vereinigung: $P \cup Q =_{df} \{ t \mid t \in P \text{ oder } t \in Q \}$

2) Durchschnitt: $P \cap Q =_{df} \{ t \mid t \text{ ist C-Tupel über } \alpha \wedge (\exists t' \in P)(\exists t'' \in Q)$
$(t|_\alpha = t'|_\alpha \wedge t(COND) = \#t'(COND) \wedge t''(COND)$
$\wedge \bigwedge_{A \in \alpha} t'(A) = t''(A)\#)\}$

Die Festlegung, daß die Variablen aus P als Attributwerte in das Ergebnis übernommen werden, ist willkürlich. Der erzeugte Ausdruck in COND sichert ab, daß diese Variablen auch durch die entsprechenden Variablen aus Q ersetzt werden könnten.

Beispiel:

U

A	B	COND
3	ω_1	

W

A	B	COND
2	ω_2	
3	ω_3	

U ∩ W

A	B	COND
3	ω_1	$\omega_1 = \omega_3$

3) Differenz:

$P \backslash Q =_{df} \{t \mid t \text{ ist C-Tupel über } \alpha \wedge (\exists t' \in P)(t|_\alpha = t'|_\alpha \wedge$
$t(COND) = \#t'(COND) \wedge \bigwedge_{t'' \in Q} (\neg t''(COND) \vee \bigvee_{A \in \alpha} t'(A) \neq t''(A))\#)\}$

Beispiele: a) Mit U und W wie oben erhalten wir als Ergebnis für den Ausdruck $U \backslash (U \cap W)$:

A	B	COND
3	ω_1	$\neg(\omega_1 = \omega_3) \vee 3 \neq 3 \vee \omega_1 \neq \omega_3$

$\equiv$

A	B	COND
3	ω_1	$\omega_1 \neq \omega_3$

b)

S

A	B	COND
1	ω_1	

T

A	B	COND
1	ω_2	

X

A	B	COND
1	ω_3	

Wir entwickeln schrittweise das Ergebnis des Ausdrucks $(S\backslash T)\backslash(S\backslash X)$:

$S\backslash T$

A	B	COND
1	ω_1	$\omega_1 \neq \omega_2$

$S\backslash X$

A	B	COND
1	ω_1	$\omega_1 \neq \omega_3$

$(S\backslash T)\backslash(S\backslash X)$

A	B	COND
1	ω_1	$\omega_1 \neq \omega_2 \wedge \omega_1 = \omega_3$

4) Projektion: Sei $\gamma \subseteq \alpha$;

$$P[\gamma] =_{df} \{t|_{\gamma \cup \{COND\}} \mid t \in P\}$$

5) Selektion: Sei vga ein Vergleichsausdruck wie bei der gewöhnlichen Selektion;

$$P[vga] =_{df} \{t \mid t \text{ ist C-Tupel über } \alpha \wedge (\exists t' \in P)(t|_\alpha = t'|_\alpha \wedge t(COND) = \#t'(COND) \wedge [t/vga]\#)\}.$$

Dabei ist [t/vga] der Ausdruck, der aus vga entsteht, indem alle Attribute in vga durch die entsprechenden Werte von t ersetzt werden.

6) Join:

$$P \bowtie S =_{df} \{t \mid t \text{ ist C-Tupel über } \alpha \cup \beta \wedge (\exists t' \in P)(\exists t'' \in S)$$
$$(t|_\alpha = t'|_\alpha \wedge t|_{\beta\setminus\alpha} = t''|_{\beta\setminus\alpha} \wedge t(COND) =$$
$$\#t'(COND) \wedge t''(COND) \wedge \bigwedge_{A \in \alpha\cap\beta} t'(A) = t''(A)\#)\}$$

Beispiel:

U

A	B	C	COND
1	ω_1	4	
2	ω_2	ω_3	

W

B	C	D	COND
3	ω_4	6	
ω_5	5	7	

$U \bowtie W$

A	B	C	D	COND
1	ω_1	4	6	$\omega_1 = 3 \wedge 4 = \omega_4$
2	ω_2	ω_3	6	$\omega_2 = 3 \wedge \omega_3 = \omega_4$
2	ω_2	ω_3	7	$\omega_2 = \omega_5 \wedge \omega_3 = 5$

Diese Operationen sind verträglich mit den Operationen der gewöhnlichen Relationenalgebra, d.h. sie liefern für totale Relationen das gleiche Ergebnis. Daher rechtfertigt sich die Verwendung derselben Operationssymbole.

Bei Auswertung eines Algebraausdrucks in einem C-Zustand z_C mit den gegebenen Operationsdefinitionen gibt in jedem Zwischenergebnis der Wert eines Tupels im Attribut COND an, für welche Variablenbelegungen, d.h. für welche möglichen Zustände zu z_C, das betreffende Tupel zum entsprechenden Zwischenergebnis beiträgt. Das Endergebnis repräsentiert somit alle möglichen Antworten.

Satz 6.4: Sei e ein beliebiger Ausdruck der Relationenalgebra vom Typ β zu einem DB-Schema σ über der Attributmenge $\alpha \subseteq \alpha_0$ (α_0 enthalte zusätzlich zu den Attributen aus α alle in e vorkommenden Attribute). Sei z_V ein V-Zustand zu σ und z_C der entsprechende C-Zustand. Sei $\mathfrak{W}$ die C-Relation, die durch Auswertung von e in z_C gemäß den angegebenen Operationsdefinitionen erhalten wird. Dann gilt:

$(\forall \nu$: ν Belegung für $\alpha_0, V)(\nu(\mathfrak{W})[\beta] = e(\nu(z_V)))$.

Der Beweis zu Satz 6.4 findet sich in [IL84]. Falls man an gesicherten uni-V-Antworten interessiert ist, ergibt sich damit folgende Definition einer Antwort auf der Basis des Endergebnisses einer Auswertung:

Definition: Mit den Bezeichnungen aus Satz 6.4 ist die C-Antwort auf e in z_C (in Zeichen: $\mathfrak{A}_C(e,z_C)$) wie folgt definiert:

$\mathfrak{A}_C(e,z_C) =_{df} \{t \mid t$ V-Tupel über $\beta \wedge (\forall \nu$: ν Belegung für $\alpha_0, V)(\nu(t) \in \nu(\mathfrak{W})[\beta])\}$

Unmittelbar aus Satz 6.4 ergibt sich

Satz 6.5: Seien e und z_C wie oben. Dann gilt:

$\mathfrak{A}_C(e,z_C) = \mathcal{GA}_{uni}(e,z_C)$.

Mit der Definition der C-Antwort ist noch kein Verfahren gegeben, wie die Antworttupel aus $\mathfrak{W}$ gewonnen werden können. In [IL84] werden nur totale Antworten betrachtet, aber auch hierfür wird kein Berechnungsverfahren angegeben. Es genügt zum Ermitteln totaler gesicherter Antworten nicht, die Tupel einzeln zu betrachten und festzustellen, ob sie im COND-Attribut einen zu *wahr* äquivalenten Ausdruck haben. Diese Vorgehensweise führt nur dann zum gewünschten Resultat, wenn zu den oben genannten Regeln eine weitere hinzukommt:

4) Falls für ein Tupel t durch t(COND) der Wert einer Variablen eindeutig bestimmt ist, ersetze die Variable in allen Attributen des Tupels, in denen sie vorkommt, durch diesen Wert.

Wir benötigen diese Regel, da wir bei der Selektionsoperation im Fall von Gleichheitsbedingungen keine eventuell mögliche Ersetzung von Variablenvorkommen durch Konstanten vornehmen. Darüberhinaus kann die Festlegung genau einer Konstanten als möglicher Wert für eine Variable auch über komplexere Ausdrücke (z.B.: $\omega_i \geq c \wedge \omega_i < c + 1$) oder indirekt über Vergleiche mit anderen Variablen erfolgen.

Beispiel 6.4: Seien $R = \{(2,2,$*wahr*$)\}$ und $S = \{(2,\omega_1,$*wahr*$)\}$ C-Relationen über {A,B}. Für $(R \setminus S) \cup S[B = 2]$ erhalten wir $\{(2,2, \omega_1 \neq 2), (2,\omega_1, \omega_1 = 2)\}$. Mit Anwendung von Regel 4 und anschließend Regel 1 ergibt sich $\{(2,2, \omega_1 \neq 2 \vee \omega_1 = 2)\}$ und damit (Regel 3) $\{(2,2,$ *wahr*$)\}$.

Wenn alle Möglichkeiten ausgeschöpft sind, die Regeln 1-4 so anzuwenden, daß möglichst wenig Tupel in der Ergebnisrelation enthalten sind, läßt sich die totale gesicherte Antwort bestimmen, indem für alle in den gewöhnlichen Attributen totalen Tupeln überpüft wird, ob der Ausdruck im COND-Attribut äquivalent zu *wahr* ist. Ist dies der Fall, dann liegt keine Abhängigkeit eines Tupels von einer Belegung der Variablen vor, d.h. das Tupel, eingeschränkt auf die gewöhnlichen Attribute, ist in jeder möglichen Antwort enthalten.

In [IL84] wird gezeigt, daß zur Überprüfbarkeit der Ausdrücke im COND-Attribut lediglich vorauszusetzen ist, daß alle Wertebereiche von Attributen, die in Vergleichsausdrücken mit den Operatoren "$\geq$" und "$\leq$" auftreten, total geordnet sind. Dies zu verlangen ist keine wesentliche Einschränkung, da andernfalls Vergleiche mit undefiniertem Ergebnis möglich sind.

Für andere Antwortformen ist das Vorgehen nicht ganz so einfach. Bevor wir dazu kommen, wollen wir noch kurz auf die Komplexitätsproblematik dieser Form einer Relationenalgebra eingehen. Es ist relativ einfach zu zeigen, daß mit den angegebenen Operationen das bekannte 3DNF-Tautologie-Problem auf das Problem zurückgeführt werden kann, festzustellen, ob eine vorgegebene Menge von totalen Tupeln in jeder möglichen Antwort enthalten ist ([AKG91]). Dabei sind in Selektionsausdrücken nur = und $\neq$ als Vergleichsoperatoren notwendig. Wegen der co-NP-Vollständigkeit des 3DNF-Problems können wir daher schon bei dieser einfachen Anwortform keine effizienten Algorithmen erwarten (ohne wieder, wie schon immer bisher, auf Vollständigkeit bei den Antworten zu verzichten). Natürlich tritt dieses Komplexitätsproblem schon bei Anwendung der Regeln 2 und 3 von oben auf.

Bemerkung: Die Notwendigkeit von Regel 4 hat keinen Einfluß auf die Komplexitätsaussage. Nach dem Raten einer Belegung kann Regel 1 angewandt werden, die nur polynomielle Kosten verursacht.

Wegen Satz 6.4 können die in Kapitel 3 eingeführten Antwortformen für jeden Algebraausdruck mit Hilfe dieser Ergebnisrelation bestimmt werden. Betrachten wir als erstes gesicherte uni-V-Antworten, die nach Satz 6.5 mit der jeweiligen C-Antwort übereinstimmen. Für ihre Bestimmung ist folgendes Lemma hilfreich:

Lemma 6.3: Für jedes $t'' \in \mathfrak{A}_C(e,z_C)$ gibt es genau ein $t \in \mathfrak{W}$ mit $t'' = t|_\beta$.

Beweis: Sei ein $t'' \in \mathfrak{A}_C(e,z_C)$ gegeben, für das in $\mathfrak{W}$ kein t existiert mit $t'' = t|_\beta$. Wegen $(\forall \nu: \nu \text{ Belegung für } \alpha_0,V)(\nu(t_i) = \nu(t_j)) \Rightarrow t_i = t_j$ für beliebige V-Tupel t_i, t_j über β gibt es dann eine Belegung ν' mit $(\forall t \in \mathfrak{W})(\nu'(t|_\beta) \neq \nu'(t''))$, d.h. $\nu'(t'') \notin \nu'(\mathfrak{W})[\beta]$. Daraus folgt aber $t'' \notin \mathfrak{A}_C(e,z_C)$ im Widerspruch zur Annahme. □

Wir können nun eine Vorgehensweise angeben, durch die gesicherte uni-V-Antworten erhalten werden (wir benötigen hier natürlich einen deterministischen Algorithmus):

- Für jedes $t \in \mathfrak{W}$ bilde folgenden Booleschen Ausdruck B_t:

$$B_t = \# \bigvee_{\{t' \in \mathfrak{W} \mid (\exists t'' \text{ über } \beta)\ (t'' \geq t|_\beta \wedge t'' \geq t'|_\beta)} (t'(COND) \wedge \bigwedge_{\{A \in \beta \mid t'(A) \neq t(A)\}} t'(A) = t(A)) \#$$

- Nimm alle $t|_\beta \in \mathfrak{W}[\beta]$ mit $B_t \equiv$ *wahr* in das Ergebnis auf.

Alle Tupel, die eine gemeinsame Überdeckung mit einem Tupel t aus $\mathfrak{W}$ haben, tragen zu dem für t gebildeten Ausdruck B_t bei. Für sie wird jeweils angegeben, unter welchen Bedingungen für die Belegungen eine Übereinstimmung mit t erhalten wird.

Beispiel 6.5: Sei folgende C-Relation W Resultat einer Auswertung:

W

A	B	C	COND
1	ω_1	ω_2	$\omega_1 \neq 3 \wedge \omega_2 = 4$
1	3	4	$\omega_1 = 3 \wedge \omega_2 = 4$
1	3	ω_2	$\omega_2 \neq 4 \vee \omega_1 \neq 3$

Wir erhalten als Bedingungsausdrücke für die einzelnen Tupel:

$(1,\omega_1,\omega_2)$: $\omega_1 \neq 3 \wedge \omega_2 = 4 \quad \vee \quad \omega_1 = 3 \wedge \omega_2 = 4 \wedge 3 = \omega_1 \wedge 4 = \omega_2 \quad \vee$
$(\omega_2 \neq 4 \vee \omega_1 \neq 3) \wedge 3 = \omega_1 \equiv$

$\omega_1 \neq 3 \wedge \omega_2 = 4 \quad \vee \quad \omega_1 = 3 \wedge \omega_2 = 4 \quad \vee \quad \omega_2 \neq 4 \wedge \omega_1 = 3 \equiv$

$\omega_2 = 4 \quad \vee \quad \omega_1 = 3$

$(1,3,4)$: $\omega_1 = 3 \wedge \omega_2 = 4 \quad \vee \quad \omega_1 \neq 3 \wedge \omega_2 = 4 \wedge \omega_1 = 3 \wedge \omega_2 = 4 \quad \vee$
$(\omega_2 \neq 4 \vee \omega_1 \neq 3) \wedge \omega_2 = 4 \equiv$

$\omega_1 = 3 \wedge \omega_2 = 4 \quad \vee \quad \omega_1 \neq 3 \wedge \omega_2 = 4 \equiv$

$\omega_2 = 4$

$(1,3,\omega_2)$: $\omega_2 \neq 4 \vee \omega_1 \neq 3 \quad \vee \quad \omega_1 \neq 3 \wedge \omega_2 = 4 \wedge \omega_1 = 3 \quad \vee$
$\omega_1 = 3 \wedge \omega_2 = 4 \wedge 4 = \omega_2 \equiv$

$\omega_2 \neq 4 \vee \omega_1 \neq 3 \quad \vee \quad \omega_1 = 3 \wedge \omega_2 = 4 \equiv$

wahr

Satz 6.6: Mit der angegebenen Vorgehensweise erhält man stets die gesicherte uni-V-Antwort.

Beweis: Wir zeigen zunächst die Korrektheit der Vorgehensweise, d.h.:

$(\forall t \in \mathfrak{W})(B_t \equiv \mathit{wahr} \Rightarrow t|_\beta \in \mathcal{GA}_{uni}(\varrho, z_V))$

Sei ein $t \in \mathfrak{W}$ mit $B_t \equiv \mathit{wahr}$ gegeben. Da B_t eine Disjunktion von Bedingungen an einzelne Tupel von $\mathfrak{W}$ darstellt, hat dann für jede Belegung mindestens einer der einzelnen Disjunktionsausdrücke den Wert *wahr*. Für eine beliebige Belegung ν sei t' ein solches Tupel. Nach Konstruktion von B_t gilt dann $\nu(t'(\mathrm{COND})) \equiv \mathit{wahr}$ und $\nu(t'|_\beta) = \nu(t|_\beta)$. Daraus folgt $\nu(t|_\beta) \in \varrho(\nu(z_V))$. Da t und ν beliebig gewählt waren, ist die Korrektheit damit bewiesen.

Für die Vollständigkeit der Vorgehensweise ist zu zeigen:

$(\forall t'' \in \mathcal{GA}_{uni}(\varrho, z_V))(\exists t \in \mathfrak{W})(t'' = t|_\beta \wedge B_t \equiv \mathit{wahr})$

Sei $t'' \in \mathcal{GA}_{uni}(e,z_V)$ und für alle $t \in \mathfrak{W}$ gelte $t'' \neq t|_\beta \vee B_t \neq \textit{wahr}$ oder äquivalent $t'' = t|_\beta \Rightarrow B_t \neq \textit{wahr}$. Wegen Lemma 6.3 gibt es zu t'' genau ein $t \in \mathfrak{W}$ mit $t'' = t|_\beta$. Für dieses t muß nach der Annahme $B_t \neq \textit{wahr}$ gelten. Damit gibt es eine Belegung ν' derart, daß für kein $t' \in \mathfrak{W}$ mit $(\exists t'')(t'' \geq t|_\beta \wedge t'' \geq t'|_\beta)$ gilt: $\nu'(t'(COND)) \equiv \textit{wahr}$ und $(\forall A \in \beta)(\nu'(t'(A)) = \nu'(t(A)))$. Ein Tupel aus $\mathfrak{W}$, das mit t keine gemeinsame Überdeckung hat, kann für keine Belegung das gleiche totale Tupel ergeben wie t. Somit kann $\nu'(t)|_\beta$ nicht in $\mathfrak{A}_C(e,z_C)$ und damit $\mathcal{GA}_{uni}(e,z_V)$ enthalten sein im Widerspruch zur Annahme. □

Folgende Frage lassen wir hier offen: Für welche Algebraausdrücke genau kann auf die Bildung von B_t für alle Tupel in $\mathfrak{W}$ verzichtet werden, d.h. wann genügt es, nur den Wert jedes Tupels im COND-Attribut zu überprüfen?

Kommen wir nun zu gesicherten Maximum-V-Antworten (im folgenden kurz max-V-Antworten). In solchen Antworten kann es notwendig sein, neue Variable zu verwenden, d.h. Variable, die im gegebenen V-Zustand nicht vorkommen. Daher können in einer solchen Antwort Tupel auftreten, die in der Ergebnisrelation $\mathfrak{W}$ nicht direkt vorhanden sind, sondern lediglich von Tupeln aus $\mathfrak{W}$ überdeckt werden. Es ergibt sich gegenüber gesicherten uni-V-Antworten die Schwierigkeit, daß für ein Tupel t aus $\mathfrak{W}$ im allgemeinen unterschiedliche Mengen von Tupeln aus $\mathfrak{W}$ für die Erzeugung eines Antworttupels betrachtet werden müssen. Die Anzahl der maximalen Elemente in $\mathcal{GT}(e,z_V)/_\equiv$ kann exponentiell sein in der Anzahl der Attribute des Typs β von e. Dies ergibt sich aus den Eigenschaften der Überdeckungsrelation: Zu einem gegebenen Tupel t gibt es höchstens $\binom{|\beta|}{\beta/2}$ Tupel, falls $|\beta|$ gerade, bzw. $\binom{|\beta|}{(|\beta|-1)/2}$ Tupel, falls $|\beta|$ ungerade, die von t überdeckt werden, aber untereinander keine Überdeckungen sind. Diese Schranke kann erreicht werden, wie das folgende Beispiel zeigt, das sich auf einfache Weise verallgemeinern läßt.

Beispiel 6.6: Sei folgende C-Relation gegeben:

W

A	B	C	COND
1	2	3	$\omega_1 \neq 1$
1	2	4	$\omega_1 \neq 2$
1	3	3	$\omega_1 \neq 3$
2	2	3	$\omega_1 \neq 4$

Für jedes Paar von Tupeln t,t' aus W ist meet({t,t'}) in der durch W bestimmten Menge aller gesicherten Antworttupel enthalten, da t(COND) ∨ t'(COND) ≡ *wahr* gilt. Als Repräsentanten für die gesicherte max-V-Antwort können wir $(1,2,\psi_1)$, $(1,\psi_2,3)$ und $(\psi_3,2,3)$ wählen. Diese Tupel erhalten wir durch Verknüpfung des ersten Tupels mit den drei anderen Tupeln. Die Verknüpfungen der anderen Tupel untereinander ergeben keine weiteren maximalen Tupel.

Da die Anzahl der zu einem Tupel aus $\mathfrak{W}$ in der Antwort möglichen Tupel mit der Attributanzahl der Antwort (exponentiell) beschränkt ist, enthält eine gesicherte max-V-Antwort zu einer festen Anfrage höchstens eine in der Größe des Zustands lineare Anzahl von Tupeln (allerdings mit einem eventuell großen konstanten Faktor).

Für die Bestimmung einer Antwort kann man sich die lineare Beschränktheit der Tupelanzahl zunutze machen, indem man über alle Tupel iteriert, die bezüglich der partiellen Ordnung $\geq_i$ von einem Tupel aus $\mathfrak{W}$ überdeckt werden, wobei die Variablen aus V wie Konstanten betrachtet werden (die Ordnungsrelations $\geq_i$ sei im folgenden geeignet geändert). Dabei muß für jede Äquivalenzklasse bzgl. $\equiv_i$ nur ein Exemplar betrachtet werden. Für jedes solche Tupel ist dann festzustellen, ob es für jede Belegung ν eine Überdeckung in $\nu(\mathfrak{W})$ hat und maximal ist bezüglich $\geq_i$. Tupel, zu denen schon eine Überdeckung im Zwischenresultat existiert, müssen nicht untersucht werden. Tupel sind aus dem Zwischenresultat wieder zu entfernen, wenn sich ein "informativeres" Tupel qualifiziert. Diese Situation kann durch geeignete Berücksichtigung von $\geq_i$ beim Iterieren vermieden werden.

Bei der Wahl von Tupeln aus den Äquivalenzklassen muß so vorgegangen werden, daß für die von einem Tupel $t \in \mathfrak{W}$ überdeckten Tupel stets Variable aus V' verwendet werden, die für kein anderes Tupel aus $\mathfrak{W}$ für die von ihm überdeckten Tupel hergenommen wurden. Dadurch wird abgesichert, daß in der Antwort keine unzulässigen Identifizierungen von Werten in verschiedenen Tupeln über Variable aus V' erfolgen. So muß in Beispiel 6.6 in jedem Tupel eine andere Variable aus V' verwendet werden. In einzelnen Tupeln können dagegen solche Variable durchaus mehrfach auftreten, um anzuzeigen, daß von diesem Tupel nur Tupel mit identischen Werten repräsentiert werden (Beispiel: (1,2,2), (1,3,3) $\rightarrow$ (1,ψ_1,ψ_1)).

Algorithmus max-V-Antwort

Gegeben: eine C-Relation $\mathfrak{W}$;

$M := \phi$;
for each $t \in \mathfrak{W}$ **do**
  for each t' **with** $t|_\beta \geq_i t'$ **and** $[\text{meet}(\{t''|_\beta \mid t'' \in \mathfrak{W} \wedge t''|_\beta \geq_i t'\})] = [t']$
    and not $(\exists m \in M)(m \geq_i t')$ **do**

$$\textbf{if} \quad \Big(\bigvee_{\{t'' \in \mathfrak{W} \mid t''|_\beta \geq_i t'\}} t''(\text{COND}) \Big) \equiv \textit{wahr}$$

$$\textbf{then} \quad M := (M \cup \{t'\}) \setminus \{t'' \in M \mid t' \geq_i t''\}$$

  end;

Für festes β ist zu jedem $t \in \mathfrak{W}$ nur eine in $|\mathfrak{W}|$ lineare Anzahl von Tupeln t' zu betrachten. Für jedes Tupel t' ist zur Überprüfung der im **with**-Teil angegebenen Bedingungen sowohl $\mathfrak{W}$ als auch M nur einmal zu durchlaufen. Die Überprüfung der Disjunktion dagegen ist, wie oben schon erwähnt, im allgemeinen vermutlich nicht effizient durchführbar. Mit diesem Ausdruck wird festgestellt, ob ein Tupel in der Menge gesicherter Antworttupel enthalten ist.

Beim Durchlaufen der von einem Tupel aus $\mathfrak{W}$ überdeckten Tupel kann systematisch wie folgt vorgegangen werden:

- Es werden die Tupel gemäß der durch $\geq_i$ gegebenen partiellen Ordnung erzeugt.
- Falls ein Tupel sich qualifiziert, werden anschließend keine von diesem Tupel überdeckten Tupel mehr erzeugt.

- Ein Tupel t' muß nicht weiter betrachtet werden, falls für die Menge $\{t'' \in \mathfrak{W} \mid t''|_\beta \geq_i t'\}$ folgendes gilt: Sei G der Graph, der sich dadurch ergibt, daß alle Tupel t'' als Knoten betrachtet werden, die miteinander über eine Kante verbunden sind, wenn in den zugehörigen COND-Werten die gleiche Variable vorkommt; dann ist G zusammenhängend.

  Begründung: Ist G nicht zusammenhängend, dann ist die anschließend gebildete Disjunktion schon für einen Teilausdruck äquivalent zu *wahr* und es gibt daher eine Überdeckung von t', die gesichertes Antworttupel ist, oder die Disjunktion kann nicht äquivalent zu *wahr* sein.

Die Korrektheit des Algorithmus ergibt sich aus folgenden Feststellungen:

1) Durch die Betrachtung der von einem Tupel aus $\mathfrak{W}[\beta]$ überdeckten Tupel gemäß der durch $\geq_i$ gegebenen partiellen Ordnung werden alle Tupel erfaßt, die maximal sind in der Menge gesicherter Antworttupel.
2) Falls ein Tupel t' die angegebene Äquivalenz erfüllt, ist es in der Menge gesicherter Antworttupel enthalten: Zu jeder Belegung ν für β,V $\cup$ V' findet sich ein Tupel $t''|_\beta \geq_i t'$ in $\mathfrak{W}$ mit $\nu(t''(COND)) \equiv$ *wahr* gemäß der Booleschen Logik. Dies folgt unmittelbar aus dem für t' gebildeten Disjunktionsausdruck.
3) Falls ein Tupel t' nicht maximal ist in der Menge gesicherter Antworttupel, wird es nicht erzeugt, da sich zuvor schon ein überdeckendes Tupel zu dem gleichen Tupel aus $\mathfrak{W}$ qualifizierte, oder es erfüllt die Bedingung $(\exists t'' \in \mathfrak{W})(t''|_\beta \geq_i t')$ nicht, oder es wird durch die letzte Anweisung beim Hinzufügen eines überdeckenden Tupels aus dem Resultat entfernt.

Wir erhalten:

Satz 6.7: Der Algorithmus zur Bestimmung einer gesicherten max-V-Antwort ist korrekt und vollständig.

Kommen wir nun zur Bestimmung einer min/max-V-Antwort bei gegebener Relation $\mathfrak{W}$. Diese Antwort enthält stets die entsprechende max-V-Antwort. Zusätzlich enthält sie Tupel, die absichern, daß von ihr jede mögliche Antwort als mögliche Relation erhalten werden kann. Die Interpretation der Variablen aus V' geschieht dabei in der gleichen Weise wie die Interpetation der Variablen aus V, d.h. über Belegungsfunktionen. Um das Vorgehen etwas zu vereinfachen, erzeugen wir in dem nachfolgend angegebenen Algorithmus zu jeder Äquivalenzklasse bezüglich $\equiv_i$, die für die Antwort benötigt wird, nur einen Repräsentanten. Das Hinzufügen der notwendigen Anzahl von Tupeln aus einer solchen Klasse ist in einfacher Weise möglich. Wenn wir im folgenden von Tupel sprechen, verstehen wir darunter stets einen beliebigen, aber fest gewählten Repräsentanten einer Klasse bezüglich $\equiv_i$.

Alle Tupel in einer min/max-V-Antwort müssen von Tupeln aus der max-V-Antwort überdeckt werden, da sie Elemente der Menge gesicherter Antworttupel sind. Damit liegt es nahe, diese Tupel ausgehend von den Tupeln der max-V-Antwort zu bestimmen. Da eine Relation $\mathfrak{W}$ alle möglichen Antworten beschreibt und zu jedem ihrer Tupel eine Überdeckung in mindestens einer möglichen Antwort existiert, kann mit Hilfe von $\mathfrak{W}$ überprüft werden, ob eine Tupelmenge alle möglichen Antworten erzeugen kann (unter eventuellem Hinzufügen äquivalenter Tupel wie oben erwähnt). Es muß dazu nur festgestellt werden, ob jedes Tupel aus $\mathfrak{W}$ eine Überdeckung für ein Tupel der Tupelmenge darstellt.

Da min/max-V-Antworten Minimalitätskriterien erfüllen müssen (Bedingungen 2 und 3 in der Definition), muß beim Erzeugen überdeckter Tupel systematisch vorgegangen werden; ansonsten ist die Überprüfung dieser Kriterien für eine gegebene Tupelmenge im allgemeinen nicht effizient möglich. Wir gehen dazu wie folgt vor: Für die Tupel der max-V-Antwort werden alle von ihnen überdeckte Tupel mit neuen Variablen nur aus V' gemäß der durch $\geq_i$ gegebenen partiellen Ordnung erzeugt. Dabei wird stufenweise vorgegangen, indem zunächst für alle Tupel der max-V-Antwort diejenigen echt überdeckten Tupel betrachtet werden, die maximal sind bezüglich $\geq_i$. Anschließend werden sukzessive von den verbleibenden überdeckten Tupeln immer die maximalen Elemente betrachtet. Die Bedingungen an die Vorkommen von Variablen aus V' in den Tupeln gewährleisten dabei, daß kein Tupel, das von mehr als einem maximalen Tupel überdeckt wird, für diese maximalen Tupel auf unterschiedlichen Stufen erreicht werden kann. Durch diesen "Abstieg" in der Menge gesicherter Antworttupel (Halbverband) wird gewährleistet, daß kein Tupel überprüft wird, für das zuvor nicht alle es überdeckenden Tupel betrachtet wurden. Wäre dies der Fall, könnte Bedingung 2 (minimale Menge möglicher Antworten) verletzt werden.

Im folgenden Algorithmus geben anz-V'(t') die Anzahl verschiedener Variablen aus V' in t' und vork-V'(t') die Anzahl der Vorkommen von Variablen aus V' in t' an. In MM wird das Ergebnis, d.h. eine min/max-V-Antwort aufgebaut. β sei wieder der Typ der Anfrage. Variablen aus V seien wie Konstanten betrachtet.

Algorithmus min/max-V-Antwort

Gegeben: max-V-Antwort M;

```
Z := {t ∈ 𝔚 | ¬ (∃m ∈ M)(t|β ≥i m)};
MM := M; j := 1;
while Z ≠ Φ do
   begin k := j;
     while k ≤ |β| do
       begin for each t' ∈ {m' | (∃m ∈ M)(m ≥i m' ∧ m ≠ m')}
                  with anz-V'(t') = j and  vork-V'(t') = k do
                if (∃z ∈ Z)(z ≥i t')
                then begin MM := MM ∪ {t'};
                           Z := Z \ {z | z ≥i t'}
                     end;
          k := k + 1
       end;
     j := j + 1
   end;
for each m ∈ MM do
   if (∀t ∈ 𝔚)(∃m' ∈ MM \{m})(t ≥i m')
   then MM := MM \ {m};
```

Die äußere **while**-Schleife sorgt dafür, daß überdeckte Tupel mit j + 1 verschiedenen Variablen aus V' erst dann betrachtet werden, wenn alle Tupel mit j verschiedenen Variablen behandelt worden sind. Die innere **while**-Schleife sichert ab, daß Überdeckungen von Tupeln mit der gleichen Anzahl verschiedener Variablen in einer Reihenfolge erzeugt werden, die nicht im Widerspruch zu $\geq_i$ steht. Als Beispiel betrachte $(1,\psi_1) \geq_i (\psi_1,\psi_1)$. Da das Tupel $(\psi_1,....,\psi_{|\beta|})$ von allen Tupeln aus $\mathfrak{W}$ über-

deckt wird, ist abgesichert, daß die äußere **while**-Schleife stets wegen Z = ϕ abbricht, ohne daß j über |β| hinausläuft. Durch die letzte **for each**-Schleife wird die Erfüllung der Bedingung 3 aus der Definition von gesicherten min/max-V-Antworten gewährleistet, indem eine minimale Menge von Tupeln bestimmt wird, die zur Erzeugung aller möglichen Antworten ausreicht.

Satz 6.8: Der Algorithmus min/max-V-Antwort liefert stets eine gesicherte min/max-V-Antwort als Ergebnis.

Die zusätzlichen Kosten, die der min/max-V-Algorithmus gegenüber dem max-V-Algorithmus bringt, sind schlimmstenfalls quadratisch in |$\mathfrak{W}$| (allerdings wiederum exponentiell in |β|).

6.3 Auswertung von TRC-Anfragen

In Kapitel 4 haben wir eine erweiterte Form der ω-Auswertung betrachtet, bei der Vorkommen von ω, die aus dem gleichen Attribut eines Tupels im Zustand stammen, identifiziert werden. Bei Auswertung von TRC-Anfragen in V-Zuständen kann eine solche Identifizierungsmöglichkeit direkt an den Variablen abgelesen werden.

Sprechweise: Um Verwechslungen mit Variablen in einem TRC-Ausdruck zu vermeiden, sprechen wir im folgenden von V-Variablen, wenn wir Variablen aus V-Zuständen bzw. der Menge V meinen.

Durch die Annahme, daß alle V-Variablen in einem Zustand verschieden voneinander sind, haben wir außerdem nicht mehr das Problem, daß zwei syntaktisch gleiche Tupel aus unterschiedlichen Relationen stammen können und voneinander unterschieden werden müssen, wenn sie nicht total sind. Daher können die in Kapitel 4 betrachteten Fallunterscheidungen bzgl. definierender und angewandter Vorkommen von Variablen in der Interpretationsvorschrift entfallen. Zum Erhalt gesicherter uni-V-Antworten genügt somit eine einfachere Vorschrift, die sich von der gewöhnlichen Interpretationsvorschrift für TRC-Ausdrücke in totalen Zuständen wie folgt unterscheidet:

- Die Belegung von Variablen erfolgt mit V-Tupeln.
- Ein Bereichsausdruck R x hat unter einer Interpretation I den Wert *wahr*, falls I(x) ∈ I(R); er hat den Wert *falsch*, falls es kein t ∈ I(R) gibt mit t Δ I(x); ansonsten hat er den Wert *unbekannt*.
- Vergleichsausdrücke der Form $\omega_i \ \Theta \ \omega_i$ erhalten den Wert *wahr*, falls Θ ∈ {≤, =, ≥}, und *falsch* sonst. Vergleichsausdrücke, in denen nur ein Operand eine Variable ist, und Vergleichsausdrücke, in denen die beiden Operanden unterschiedliche Variable sind, erhalten den Wert *unbekannt*.
- Die Auswertung Boolescher Ausdrücke erfolgt mit der Kleene'schen dreiwertigen Logik.

Sei das Ergebnis der Auswertung einer erlaubten TRC-Anfrage q im V-Zustand z_V gemäß der festgelegten Interpretationsvorschrift mit $\mathfrak{A}_V(q,z_V)$ bezeichnet.

Satz 6.9: Seien q = (x)/ Φ(x) eine erlaubte TRC-Anfrage an ein DB-Schema σ und z_V ein V-Zustand zu σ. Dann gilt

$$\mathfrak{A}_V(q,z_V) \subseteq \mathcal{GA}_{uni}(q,z_V).$$

Der Beweis des Satzes läßt sich in Anlehnung an den entsprechenden Beweis für die ω-Interpretation führen und soll daher entfallen.

Ähnlich wie wir in Kapitel 4 und 5 sogenannte sub-Auswertungen eingeführt haben, um den Informationsgehalt von Antworten zu verbessern, können auch für die Auswertung von TRC-Anfragen in V-Zuständen entsprechende Auswertungsverfahren angegeben werden. Da jede Variable genau einen unbekannten Wert repräsentiert, läßt sich die Vorgehensweise für Vervollständigungen als POSS-Funktionen für ω-Zustände (Kapitel 4) direkt übertragen. Zusätzlich kann, wie schon oben angegeben, das Wissen über die Gleichheit unbekannter Werte ausgenutzt werden. Wir wollen ein entsprechendes Verfahren wegen seiner Ähnlichkeit mit den schon vorgestellten Verfahren hier nicht weiter betrachten, sondern uns mit einer anderen Auswertungsmethode beschäftigen, die sich an C-Relationen orientiert.

Durch die Algebraoperationen für C-Relationen wird im Attribut COND jedes Tupels vermerkt, unter welchen Bedingungen für Belegungen von Variablen das betreffende Tupel im Ergebnis enthalten ist. Übertragen wir diese Idee auf die Auswertung von TRC-Ausdrücken, dann bedeutet dies, daß Vergleichsausdrücke mit Variablen als Operanden, denen kein Boolescher Wert zugewiesen werden kann, nicht zu *unbekannt* ausgewertet werden dürfen, sondern die im Vergleich enthaltene Bedingung an Variable gemerkt werden muß. Dies führt unmittelbar zu einer Auswertungsmethode, die wir "lazy evaluation" nennen wollen (in Anlehnung an die Bezeichnung entsprechender Verfahren bei Programmiersprachen).

Lazy evaluation

Sei q = (x)/ Φ(x) eine erlaubte TRC-Anfrage und I_L eine passende Interpretation. Bei der Auswertung von $I_L(\Phi(x))$ werden alle Bereichsausdrücke R x ersetzt durch

$$\bigvee_{t \in I_L(R)} \quad \bigwedge_{A \in Attr(R)} I_L(x)(A) = t(A).$$

Allen durch Belegung von Variablen erhaltenen Vergleichsausdrücken der Form

(*) $\omega_i \ominus c$, $c \ominus \omega_i$ und $\omega_i \ominus \omega_j$, $\quad i \neq j$, $\ominus \in \{<, \leq, =, \neq, \geq, >\}$

wird kein Wert zugewiesen.
Der erhaltene Ausdruck wird mit der zweiwertigen Booleschen Logik ausgewertet, als ob diese Vergleichsausdrücke Boolesche Variable wären.
Als Ergebnis erhalten wir für jede Interpretation einen Booleschen Wert oder einen Booleschen Ausdruck über Vergleichsausdrücken der Form (*). Das Ergebnis $\mathfrak{A}_L(q,z_V)$ der Auswertung von q ist dann gegeben als

$$\mathfrak{A}_L(q,z_V) =_{df} \{(I_L(x) \bowtie I_L(\Phi(x))) \mid I_L(\Phi(x)) \neq \mathit{false}\},$$

dabei soll $I_L(\Phi(x))$ den Ausdruck bezeichnen, der sich unter I_L bei Anwendung der angegebenen Interpretationsvorschrift ergibt.

Mit dieser Vorgehensweise erhalten wir wie bei der Algebra für C-Relationen eine Relation über β ∪ COND als Ergebnis, aus der die verschiedenen Formen gesicherter Antworten gewonnen werden können. Dabei sei COND wieder das Attribut, in dem die Bedingung für jedes Tupel, d.h. hier $I_L(\Phi(t))$ für jedes t, angegeben ist. Der Beweis hierfür ist allerdings sehr viel einfacher.

Satz 6.10: Sei $q = (x)/\ \Phi(x)$ eine erlaubte TRC-Anfrage vom Typ β an ein DB-Schema σ im V-Zustand z_V. Dann gilt

$(\forall \nu$: ν Belegung für $\alpha,V)(\nu(\mathfrak{A}_L(q,z_V))[\beta] = q(\nu(z_V)))$.

Beweis: Sei eine beliebige Belegung für α,V gegeben. Es gilt

$t \in \nu(\mathfrak{A}_L(q,z_V))[\beta] \Leftrightarrow (\exists t' \in \mathfrak{A}_L(q,z_V))(\nu(t')|_\beta = t \wedge \nu(t'(COND)) \equiv \mathit{wahr})$.

Bei der Bildung von $\mathfrak{A}_L(q,z_V)$ werden nur Umformungen vorgenommen, die in allen möglichen Zuständen gültig sind. Daher gilt

$t'(COND) \equiv \Phi(t'|_\beta)$ in z_V,

d.h. für jedes ν:

$\nu(t'(COND)) = \nu(\Phi(t'|_\beta))$ in $\nu(z_V)$.

Die durch eine Belegung ν gegebene Zuordnung von Variablen aus V zu Konstanten ist für in Φ gebundene Variable ohne Bedeutung, wenn Φ in $\nu(z_V)$ ausgewertet wird. Daher erhalten wir

$\nu(t'(COND)) \equiv \Phi(\nu(t'|_\beta))$ in $\nu(z_V)$

und somit

$$\begin{aligned} \nu(t'(COND)) \equiv \mathit{wahr} &\Leftrightarrow \nu(t'|_\beta) \in q(\nu(z_V)) \\ &\Leftrightarrow \nu(t')|_\beta \in q(\nu(z_V)) \\ &\Leftrightarrow t \in q(\nu(z_V)). \end{aligned}$$

□

Bei den für Algebraausdrücke erhaltenen $\mathfrak{W}$-Relationen ist es im allgemeinen notwendig, für jedes Tupel t den Ausdruck B_t zu bilden, um die gesicherte uni-V-Antwort zu erhalten. Das folgende Beispiel zeigt, daß es auch bei der "lazy evaluation" im allgemeinen erforderlich ist, überdeckende und überdeckte Tupel mit zu berücksichtigen.

Beispiel 6.7: Sei folgende V-Relation R gegeben:

R

A	B	C
1	ω_1	4
2	3	4

Betrachte die TRC-Anfrage

$$(x)/\ (\exists y)(R\ y \wedge x.A = y.B \wedge x.B = y.C \wedge (y.A = 1 \wedge y.B = 3 \vee (\exists z)(R\ z \wedge z.B \neq 3 \wedge z.A \neq y.A)))$$

Die "lazy evaluation" der Anfrage ergibt für die Belegung von x mit $(\omega_1,4)$:

$$\omega_1 =_d \omega_1 \wedge 4 =_d 4 \wedge (1 = 1 \wedge \omega_1 = 3 \vee (\omega_1 \neq 3 \wedge 1 \neq 1 \vee 3 \neq 3 \wedge 2 \neq 1)) \vee \omega_1 =_d 3 \wedge 4 =_d 4 \wedge \ldots\ldots \equiv$$

$$\omega_1 = 3$$

$=_d$ kennzeichnet Vergleiche, die aus definierenden Ausdrücken entstanden sind.
Für die Belegung von x mit (3,4) erhalten wir $\omega_1 \neq 3$ als Bedingung. Damit ist B_t

für $t = (3,4, \omega_1 \neq 3)$ eine Tautologie und die gesicherte uni-V-Antwort besteht aus dem Tupel (3,4).
Die zu Beginn des Abschnitts angegebene Interpretationsvorschrift liefert eine leere Antwort für die Anfrage.

Für die Bestimmung anderer Antwortformen kann wie bei der Relationenalgebra vorgegangen werden.

6.4 Äquivalenz von Relationenalgebra über C-Relationen und TRC mit "lazy evaluation"

Wie wir gesehen haben, führen die Operationen für C-Relationen und die "lazy evaluation" zu C-Relationen, die alle möglichen Antworten für den gegebenen Algebraausdruck bzw. die gegebene TRC-Anfrage repräsentieren (Sätze 6.4 und 6.10). Einziger Unterschied ist hierbei, daß im Fall von Algebraausdrücken Belegungen für α_O,V betrachtet werden müssen, die auch eventuell notwendige Umbenennungen mit berücksichtigen (daher die erweiterte Attributmenge α_O). Dies spielt aber weiter keine Rolle, da die Variablenmenge V_A eines Attributes A, das durch Umbenennung aus einem anderen Attribut B erhalten wurde, mit der Variablenmenge V_B übereinstimmen muß. Wir können demnach o.B.d.A. die Belegungen in den beiden genannten Sätzen identifizieren.

Die Menge der möglichen Antworten zu einer Anfrage in einem V-Zustand z_V ist über die möglichen Zustände zu z_V, d.h. über totale Zustände, festgelegt. Sind eine TRC-Anfrage q und eine Algebraanfrage e für totale Zustände äquivalent, dann sind sie daher aufgrund der Sätze 6.4 und 6.10 auch für V-Zustände äquivalent, d.h. es gilt:

Satz 6.11: Der Äquivalenzsatz für die Relationenalgebra und den TRC läßt sich von totalen Zuständen auf V-Zustände übertragen, wenn für Algebraausdrücke die Operationen für C-Relationen verwendet werden und für TRC-Anfragen die "lazy evaluation".

6.5 Beweistheoretische Sichtweise

Werden relationale Datenbanken im Unterschied zur üblichen modelltheoretischen Sichtweise als logische Theorien betrachtet ([NG78], [GaMi78], [Rei80], [Rei84]), dann spricht man auch von einer *beweistheoretischen Sichtweise* von Datenbanken ([Rei84]). Diese Betrachtungsweise erlaubt wegen ihrer Beschreibung von DB-Zuständen über Axiome eine elegante Einführung unbekannter Werte: Die Menge der Konstantensymbole wird als disjunkte Vereinigung zweier Mengen C und Ω betrachtet. Zu jedem Paar unterschiedlicher Konstantensymbole $c_i, c_j \in C$ gibt es ein Eindeutigkeitsaxiom der Form $\neg E(c_i,c_j)$ oder $\neg E(c_j,c_i)$. Dabei steht E für die Gleichheit. Für Elemente aus Ω kann es ebenfalls solche Eindeutigkeitsaxiome geben; es gibt aber keine Bedingungen für die Existenz solcher Axiome. Tritt ein Konstantensymbol aus Ω in keinem Eindeutigkeitsaxiom auf, dann entspricht dieses Symbol einer Variablen im Sinne der V-Relationen, da für es nicht mehr festgestellt werden kann, ob es von beliebigen anderen Konstantensymbolen (aus C oder aus Ω) verschieden ist. Alle übrigen Axiome (Abschlußaxiome für Wertebereiche, Vollständigkeitsaxiome für Prädikate und die für die Formulierung dieser Axiome notwendigen Gleichheitsaxiome) bleiben unverändert.

Die modelltheoretische und die beweistheoretische Sichtweise von Datenbanken können unter der Annahme endlicher Wertebereiche für totale DB-Zustände als äquivalent angesehen werden ([Rei84]). Aus beweistechnischen Gründen ist es für Vergleiche einfacher, auf der modelltheoretischen Seite den wertebereichsorientierten Kalkül (DRC) an Stelle des TRC zu betrachten. Endliche Wertebereiche stellen insofern keine echte Beschränkung dar, da DB-Zustände stets als endlich angenommen werden und die eventuell unendlichen Wertebereiche zu Attributen wegen der CWA in Verbindung mit der Beschränkung auf wertebereichsunabhängige Anfragen wie endliche Wertemengen gesehen werden können. Aufgrund der Äquivalenz beider Sichtweisen sind für die Bestimmung gesicherter Antworten in partiellen DB-Zuständen mit der beweistheoretischen Sichtweise keine effizienten Verfahren zu erwarten, wo auch mit der modelltheoretischen Sichtweise keine Effizienz erreichbar ist.

Im folgenden untersuchen wir einige Zusammenhänge, die sich zwischen unserer Vorgehensweise und Vorschlägen für Antworten und Auswertungsverfahren im Fall der beweistheoretischen Sichtweise ergeben. Dabei argumentieren wir nicht immer streng formal, um uns möglichst wenig mit rein technischen Details beschäftigen zu müssen. Insbesondere übertragen wir Feststellungen zum TRC ohne weitere Erläuterung auf den (äquivalenten) DRC. Aus den Betrachtungen folgen die behaupteten Zusammenhänge aber zwingend.

Es sei darauf hingewiesen, daß bei der Angabe von Verfahren zur Auswertung von Anfragen in [Rei86] und [YC88] eine etwas geänderte Betrachtungsweise von "Nullwerten" zugrunde gelegt wird, die nicht die Bedeutung von unbekannten Attributwerten im Sinne von V-Relationen und V-Zuständen trifft. Es wird dort bei den Konstantensymbolen nicht zwischen "gewöhnlichen" Symbolen und "Nullwerten" unterschieden. Diese Unterscheidung ist aber notwendig, da sie allein aus den Eindeutigkeitsaxiomen nicht ableitbar ist (siehe auch entsprechende Bemerkungen in [Lak89]). Wir beziehen uns in den folgenden Definitionen daher auf [Rei80] und [Rei84], benutzen aber auch vereinfachende Schreibweisen aus [Rei86]. Außerdem passen wir einige Begriffe an unsere Sprechweise an.

6.5.1 Eine erweiterte relationale Theorie

Eine relationale Sprache $\mathcal{R} = (\mathcal{A}, \mathcal{F})$ ist eine Sprache der Prädikatenlogik 1. Stufe (kurz: PL1-Sprache), wobei $\mathcal{A}$ ein Alphabet von Symbolen ist und $\mathcal{F}$ die Menge wohlgeformter Formeln bezeichnet, die mit den Symbolen aus $\mathcal{A}$ in der üblichen Weise konstruiert werden können. $\mathcal{A}$ setzt sich zusammen aus Variablen, Konstanten- und Prädikatensymbolen sowie den bekannten Zeichen, die für den Aufbau von PL1-Formeln benötigt werden. Da Konstantensymbole nicht weiter (d.h. als sie selbst) interpretiert werden, sprechen wir im folgenden kurz von Konstanten. Von $\mathcal{A}$ werden zusätzlich folgende Eigenschaften gefordert:

1) Es gibt nur endlich viele Konstanten und Prädikatensymbole in $\mathcal{A}$.
2) Unter den Prädikatensymbolen von $\mathcal{A}$ gibt es ein ausgezeichnetes Symbol E, das für die Gleichheit benutzt wird.
3) Unter den Symbolen von $\mathcal{A}$ gibt es eine ausgezeichnete nichtleere Teilmenge einstelliger Prädikatensymbole, sogenannte einfache Typen. Sie werden zur Modellierung des Wertebereichkonzepts verwendet.

Bemerkung: Im Unterschied zu [Rei84] beschränken wir uns hier auf einfache Typen, da wir wie in [Rei86] nur Anfragen mit Variablen betrachten wollen, deren Typ einfach ist. Dies entspricht der Zuordnung von Variablen zu Attributen im DRC.

In einer relationalen Sprache $(\mathcal{A}, \mathcal{F})$ ist ein Term eine Variable oder eine Konstante aus $\mathcal{A}$. Falls P ein n-stelliges Prädikatensymbol von $\mathcal{A}$ ist und $t_1,....,t_n$ Terme sind, ist $P(t_1,....,t_n)$ eine atomare Formel. Es seien folgende Abkürzungen in der Schreibweise von Formeln eingeführt:

$(x/\tau)(\Phi)$ steht für $(\forall x)(\tau(x) \Rightarrow \Phi)$
$(\exists x/\tau)(\Phi)$ steht für $(\exists x)(\tau(x) \wedge \Phi)$

Wie beim TRC bzw. DRC handelt es sich hier um typbeschränkte Quantoren. Falls $\mathbf{x} = x_1,....,x_n$ eine Folge unterschiedlicher Variablen ist, wird auch $\Phi(\mathbf{x})$ statt $\Phi(x_1,....,x_n)$ und $(\mathbf{x})\Phi(\mathbf{x})$ statt $(x_1,....,x_n)\Phi(x_1,....,x_n)$ geschrieben. In ähnlicher Weise kürzt $E(\mathbf{s},\mathbf{t})$ den Ausdruck $E(s_1,t_1) \wedge .... \wedge E(s_n,t_n)$ und $\neg E(\mathbf{s},\mathbf{t})$ den Ausdruck $\neg E(s_1,t_1) \vee .... \vee \neg E(s_n,t_n)$ ab, falls $\mathbf{s} = s_1,....,s_n$ und $\mathbf{t} = t_1,....,t_n$ gleichlange Folgen von Termen sind.

Definition: Sei $(\mathcal{A}, \mathcal{F})$ eine relationale Sprache, und seien die Konstanten aus $\mathcal{A}$ als disjunkte Vereinigung zweier Mengen C und Ω gegeben. Eine endliche Teilmenge $\mathcal{T} \subseteq \mathcal{F}$ ist eine erweiterte relationale Theorie mit Nullwerten $\Leftrightarrow_{df}$ $\mathcal{T}$ erfüllt die folgenden Bedingungen:

1) Für jedes von E verschiedene n-stellige Prädikatensymbol P aus $\mathcal{A}$ (also einschließlich der einfachen Typen) enthält $\mathcal{T}$ genau eine Formel der Form

   $(\mathbf{x})P(\mathbf{x}) \equiv E(\mathbf{x},\mathbf{c}^{(1)}) \vee .... \vee E(\mathbf{x},\mathbf{c}^{(r)})$,

   wobei die $\mathbf{c}^{(i)}$ n-Tupel von Konstanten aus $\mathcal{A}$ sind. Dabei ist $r = 0$ erlaubt. In diesem Fall lautet die Formel $(\mathbf{x}) \neg P(\mathbf{x})$.
   Diese Formel heißt Ausdehnungsaxiom von P in $\mathcal{T}$.
2) Für jedes Paar unterschiedlicher Konstanten c,c' aus C enthält $\mathcal{T}$ ein Eindeutigkeitsaxiom für Namen der Form $\neg E(c,c')$ oder $\neg E(c',c)$.
3) $\mathcal{T}$ kann zusätzlich Axiome der folgenden Form enthalten:

   $\neg E(c,\omega_i)$, $\neg E(\omega_i,\omega_j)$ für $c \in C$, $\omega_i,\omega_j \in \Omega$, $i \neq j$.
4) $\mathcal{T}$ enthält sonst nichts.

Bemerkung: Es handelt sich bei dieser Theorie um eine axiomatisierbare Theorie mit einem endlichen Axiomensystem.

Im Unterschied zu [Rei84] gibt es hier kein allgemeines Abschlußaxiom mit allen Konstanten, sondern Ausdehnungsaxiome für die einfachen Typen, die zusammengenommen als Abschlußaxiom betrachtet werden können. Außerdem haben wir die üblichen *Gleichheitsaxiome* (Reflexivität, Kommutativität, Transitivität und das Leibniz-Prinzip der Ersetzung gleicher Terme) nicht mit aufgenommen, da sie in jeder relationalen Theorie vorhanden sein müssen. Der wesentliche Unterschied zur Definition einer gewöhnlichen relationalen Theorie besteht in der Unterteilung der Konstantenmenge in gewöhnliche Konstanten und Konstanten, die Platzhalter für unbekannte Werte darstellen sollen. In einer gewöhnlichen relationalen Theorie muß für jedes Paar verschiedener Konstanten c, c' aus $\mathcal{A}$ ein Axiom $\neg E(c,c')$

vorhanden sein. In einer erweiterten relationalen Theorie müssen nicht alle Konstanten voneinander unterscheidbar sein, sondern nur diejenigen aus der Menge C. Tritt eine Konstante aus Ω in keinem Eindeutigkeitsaxiom auf, dann entspricht sie einer Variablen in einem V-Zustand: Sie kann von keiner anderen Konstanten aus dem Wertebereich unterschieden werden.

Durch geeignete Angabe von Eindeutigkeitsaxiomen für Elemente aus Ω kann auch disjunktive Information modelliert werden ("t hat in A den Wert a oder b"), oder es kann festgelegt werden, daß von Werten in Attributen von Tupeln lediglich bekannt ist, daß sie verschieden voneinander sind.

Beispiel 6.8: Sei eine geeignete relationale Sprache $\mathcal{R}$ gegeben mit τ_1 und τ_2 als einfachen Typen. Dann ist durch die folgende Axiomenmenge eine relationale Theorie zu $\mathcal{R}$ bestimmt:

$(x)\tau_1(x) \equiv E(x,1) \vee E(x,2) \vee E(x,\omega_2)$
$(x)\tau_2(x) \equiv E(x,2) \vee E(x,3) \vee E(x,\omega_1) \vee E(x,\omega_3) \vee E(x,\omega_4)$
$(x_1,x_2)S(x_1,x_2) \equiv E(x_1,1) \wedge E(x_2,2) \vee E(x_1,1) \wedge E(x_2,3) \vee E(x_1,2) \wedge E(x_2,\omega_1)$
$(x_1,x_2)T(x_1,x_2) \equiv E(x_1,\omega_2) \wedge E(x_2,2)$
$(x_1,x_2)U(x_1,x_2) \equiv E(x_1,1) \wedge E(x_2,\omega_3) \vee E(x_1,2) \wedge E(x_2,\omega_4)$
$\neg E(1,2),\ \neg E(1,3),\ \neg E(2,3)$

Ein entsprechender V-Zustand enthält folgende V-Relationen über den Attributen A und B, die den einfachen Typen τ_1 und τ_2 entsprechen:

S: $\{(1,2), (1,3), (2,\omega_1)\}$

T: $\{(\omega_2,2)\}$

U: $\{(1,\omega_3), (2,\omega_4)\}$

Da Anfragen in festen DB-Zuständen ausgewertet werden und DB-Zustände mit der beweistheoretischen Sichtweise als Menge von Axiomen gegeben sind, ist wegen der stets angenommenen Endlichkeit von DB-Zuständen auch die Axiomenmenge endlich. Im Vorschlag einer Anfragesprache von Reiter wird jeder Konstanten c_i ein eindeutiger Typ τ_i zugeordnet, der durch ein einstelliges Prädikat festgelegt ist (wir beziehen uns hier auf [Rei86]; in [Rei80] und [Rei84] werden - wie oben schon erwähnt - Boolesche Ausdrücke über einfachen Typen als Typen betrachtet).

Definition: Eine Anfrage ist ein Ausdruck der Form

$$\langle x_1/\tau_1,....,x_n/\tau_n \mid \Phi(x_1,....,x_n)\rangle,$$

wobei $\Phi(x_1,....,x_n)$ ein Element einer Menge wohlgeformter Formeln ist, die mit den Symbolen einer gegebenen relationalen Sprache $\mathcal{R} = (\mathcal{A},\mathcal{F})$ konstruiert werden können.

Da $\mathcal{R}$ nur über endlich viele Konstanten verfügt, ergibt sich eine Entsprechung zur Auswertung von TRC- (DRC-) Anfragen bei beschränktem Universum.

Definition: Ein n-Tupel $\mathbf{c} = (c_1,....,c_n)$ von Konstanten aus $\mathcal{R}$ ist ein Antworttupel für eine Anfrage $\langle x_1/\tau_1,....,x_n/\tau_n \mid \Phi(x_1,....,x_n)\rangle$ in einem passenden DB-Zustand z, falls $\tau_1(c_1) \wedge .... \wedge \tau_n(c_n)$ und $\Phi(\mathbf{c})$ mit der z bestimmenden Axiomenmenge $\mathcal{T} \subseteq \mathcal{F}$

beweisbar sind. Wegen der Endlichkeit der Konstantenmenge sind nur endlich viele Tupel zu überprüfen, um die Antwort, d.h. die Menge der Antworttupel, zu bestimmen. Die äquivalente modelltheoretische Festlegung lautet: Ein n-Tupel $\mathbf{c} = (c_1,....,c_n)$ von Konstanten aus $\mathcal{R}$ ist ein Antworttupel für die Anfrage $<x_1/\tau_1,....,x_n/\tau_n \mid \Phi(x_1,....,x_n)>$ in einem passenden DB-Zustand z ($\mathcal{T}$), falls die Formeln $\tau_i(c_i)$, i = 1,....,n, und $\Phi(\mathbf{c})$ *wahr* sind für jedes Modell von z ($\mathcal{T}$) in der PL1-Logik mit Gleichheit.

Wegen der durch $\tau_1(c_1) \wedge .... \wedge \tau_n(c_n)$ erfolgenden Typüberprüfung können nur solche Konstanten aus $\mathcal{A}$ in Antworten auftreten, die auch im Zustand vorkommen. Unter der Bedingung, daß alle Konstanten, die in Ausdehnungsaxiomen für einfache Typen vorkommen, auch in Ausdehnungsaxiomen für gewöhnliche Prädikate vorkommen, können wir damit eine Entsprechung zu gesicherten uni-V-Antworten erhalten, in denen nur Variable aus dem betrachteten Zustand auftreten können (im Unterschied zu den erweiterten Antwortformen, die wir definiert haben): Über die Ausdehnungsaxiome wird gewährleistet, daß Konstanten ohne Eindeutigkeitsaxiome wie alle anderen Konstanten als Belegungen von Variablen verwendet werden dürfen. Falls alle Eindeutigkeitsaxiome für eine Konstante fehlen, ist kein Wissen darüber vorhanden, ob sie mit einer der übrigen Konstanten übereinstimmt oder von allen verschieden ist. Aus dem Reflexivitätsaxiom ergibt sich, daß alle Vorkommen einer Konstanten für den gleichen Wert stehen.

6.5.2 Ein korrektes, aber unvollständiges Auswertungsverfahren

In [Rei86] wird ein Auswertungsverfahren für Anfragen vorgeschlagen, das korrekt, aber nicht vollständig ist, d.h. es werden nur Tupel als Ergebnis geliefert, die Antworttupel darstellen, allerdings werden im allgemeinen nicht alle Antworttupel erhalten. Der Grund für die Unvollständigkeit liegt darin, daß durch das Verfahren nicht alle Tautologien erfaßt werden können. Das Verfahren übersetzt eine Anfrage in einen Algebraausdruck mit sogenannten *primitiven Anfragen* als Operanden. Diese Übersetzung erfolgt ähnlich wie die Übersetzung von DRC-Anfragen in Algebraausdrücke ([Mai83]).

Primitive Anfragen haben die Form $<\mathbf{x}/\tau \mid P(\mathbf{r})>$ oder $<\mathbf{x}/\tau \mid \neg P(\mathbf{r})>$. Dabei sind P ein beliebiges n-stelliges Prädikatensymbol und $\mathbf{r}$ ein n-Tupel aus Konstanten und Variablen derart, daß alle Variablen von $\mathbf{x}$ in $\mathbf{r}$ vorkommen. Für die Festlegung der Bedeutung von primitiven Anfragen können wir auf Begriffe zurückgreifen, die wir für V-Relationen eingeführt haben.

Sei $<\mathbf{x}/\tau \mid P(\mathbf{r})>$ eine primitive Anfrage in einem DB-Zustand, der durch eine relationale Theorie $\mathcal{T}$ gegeben ist. Sei μ eine Funktion von der gleichen Art wie eine Belegungsfunktion mit dem einzigen Unterschied, daß durch sie Variable aus $\mathcal{A}$ belegt werden, wobei die einfachen Typen die Bedeutung von Attributen übernehmen. Dann ist die Antwort $\|<\mathbf{x}/\tau \mid P(\mathbf{r})>\|$ wie folgt definiert:

$$\|<\mathbf{x}/\tau \mid P(\mathbf{r})>\| =_{df} \{\mu(\mathbf{x}) \mid \mu \mid \mathbf{x} \rightarrow DOM_\tau \wedge \mu(\mathbf{r}) \in |P|\}.$$

Dabei bezeichnet DOM_τ die Menge aller Tupel über τ, die gemäß den für die Typen in τ angegebenen Ausdehnungsaxiomen gebildet werden können. $|P|$ bezeichnet die Menge aller durch das Ausdehnungsaxiom für P in $\mathcal{T}$ bestimmten Tupel zu P.

Sei $<\mathbf{x}/\tau \mid \neg P(\mathbf{r})>$ eine primitive Anfrage, wobei P nicht das Gleichheitsprädikat ist. Dann gilt

$$\mathbf{c} \in \|<\mathbf{x}/\tau \mid \neg P(\mathbf{r})>\| \Leftrightarrow_{df} \mathbf{c} \in DOM_\tau \wedge \mu(\mathbf{x}) = \mathbf{c} \Rightarrow \mu(\mathbf{r}) \notin |P|.$$

Im Fall des Gleichheitsprädikates sind andere Festlegungen zu treffen. Für unsere Betrachtungen benötigen wir diese Definitionen aber nicht.

Beispiel 6.9: Sei die gleiche Theorie wie in Beispiel 6.8 gegeben. Dann gilt:

$\|\langle y/\tau_2 \mid S(2,y)\rangle\| = \{\omega_1\}$

$\|\langle x/\tau_1 \mid S(x,x)\rangle\| = \Phi$

$\|\langle y/\tau_2 \mid \neg S(2,y)\rangle\| = \Phi$

$\|\langle x/\tau_1, y/\tau_2 \mid \neg S(x,y)\rangle\| = \Phi$

$\|\langle x/\tau_1 \mid \neg S(x,x)\rangle\| = \{(1,1)\}$

Bei der Umformung von Anfragen der Form $\langle \mathbf{x}/\tau \mid \Phi(\mathbf{r})\rangle$ in Ausdrücke mit algebraischen Operationen, die primitive Anfragen als Operanden haben, wird in [Rei86] wie folgt vorgegangen:

- Φ wird zunächst mit Hilfe bekannter Umformungsregeln in eine Form gebracht, in der nur die Junktoren $\vee$, $\wedge$ und $\neg$ vorkommen und in der das Negationszeichen nur unmittelbar vor einer atomaren Formel auftritt.
- Anschließend wird dieser Ausdruck unter Verwendung der folgenden Gleichungen bzw. Inklusionen zielgerichtet umgeformt:

1) $\|\langle \mathbf{x}/\tau \mid \Phi_1(\mathbf{x}) \wedge \Phi_2(\mathbf{x})\rangle\| = \|\langle \mathbf{x}/\tau \mid \Phi_1(\mathbf{x})\rangle\| \cap \|\langle \mathbf{x}/\tau \mid \Phi_2(\mathbf{x})\rangle\|$

2) $\|\langle \mathbf{x}/\tau \mid \Phi_1(\mathbf{x}) \vee \Phi_2(\mathbf{x})\rangle\| \supseteq \|\langle \mathbf{x}/\tau \mid \Phi_1(\mathbf{x})\rangle\| \cup \|\langle \mathbf{x}/\tau \mid \Phi_2(\mathbf{x})\rangle\|$

3) Falls $|\Theta| = \{\}$:

$\|\langle \mathbf{x}/\tau \mid (y/\Theta)\Phi(\mathbf{x},y)\rangle\| = DOM_\tau$

falls $|\Theta| \neq \{\}$:

$\|\langle \mathbf{x}/\tau \mid (y/\Theta)\Phi(\mathbf{x},y)\rangle\| = \Delta_\Theta \|\langle \mathbf{x}/\tau, y/\Theta \mid \Phi(\mathbf{x},y)\rangle\|$

4) $\|\langle \mathbf{x}/\tau \mid (\exists y/\Theta)\Phi(\mathbf{x},y)\rangle\| \supseteq \Pi \|\langle \mathbf{x}/\tau, y/\Theta \mid \Phi(\mathbf{x},y)\rangle\|$

Dabei sind Δ_Θ und Π wie folgt definiert:

$\Delta_\Theta S =_{df} \{\mathbf{a} \mid \mathbf{a}b \in S \text{ für alle } b \in |\Theta|\}$

$\Pi S =_{df} \{(a_1,\ldots,a_{n-1}) \mid (a_1,\ldots,a_{n-1},a_n) \in S\}$

5) Sei y eine Variable, die in $\Phi(x)$ nicht frei vorkommt;

a) $\|\langle y/\Theta, \mathbf{x}/\tau \mid \Phi(\mathbf{x})\rangle\| = |\Theta| \times \|\langle \mathbf{x}/\tau \mid \Phi(\mathbf{x})\rangle\|$

b) $\|\langle \mathbf{x}/\tau, y/\Theta, \mathbf{z}/\varphi \mid \Phi(\mathbf{x},\mathbf{z})\rangle\| = \Pi_{2,\ldots,n+1,1,n+2,\ldots,n+k+1}(|\Theta| \times \|\langle \mathbf{x}/\tau, \mathbf{z}/\varphi \mid \Phi(\mathbf{x},\mathbf{z})\rangle\|)$

wobei $\mathbf{x}/\tau = x_1/\tau_1,\ldots,x_n/\tau_n$, $n \geq 1$, und $\mathbf{z}/\varphi = z_1/\varphi_1,\ldots,z_k/\varphi_k$, $k \geq 0$; Π ist die Projektionsoperation mit den Stellen, auf die projiziert wird, als Index.

Die beiden Inklusionen geben an, daß die Anfrageäquivalenz bei der Übersetzung an zwei Stellen verlorengehen kann:

1) Ersetzung einer Disjunktion durch eine Vereinigung:

$\langle \mathbf{x}/\tau \mid \Phi_1(\mathbf{x}) \vee \Phi_2(\mathbf{x})\rangle \rightarrow \langle \mathbf{x}/\tau \mid \Phi_1(\mathbf{x})\rangle \cup \langle \mathbf{x}/\tau \mid \Phi_2(\mathbf{x})\rangle$

Betrachte etwa die Anfrage $<z/\tau \mid \neg E(z,a) \vee \neg E(z,b)>$, und sei $\neg E(a,b)$ ein Eindeutigkeitsaxiom der gegebenen Theorie. Sei ferner $|\tau| = \{a,b,c\}$. Dann ist $\{a,b,c\}$ die Antwort. Für $<z/\tau \mid \neg E(z,a)> \cup <z/\tau \mid \neg E(z,b)>$ erhalten wir dagegen $\{b\} \cup \{a\}$ als Antwort.

2) Umformung eines Ausdrucks mit einem führenden Existenz-Quantor durch einen Ausdruck mit einer Projektion statt dem Quantor:

$$<\mathbf{x}/\tau \mid (\exists y/\Theta)\Phi(\mathbf{x},y)> \quad \rightarrow \quad \Pi<\mathbf{x}/\tau,\ y/\Theta \mid \Phi(\mathbf{x},y)>$$

Betrachte wieder die Theorie aus Beispiel 6.8, für die gilt:

$|\tau_1| = \{1,2,\omega_2\}$, $|\tau_2| = \{2,3,\omega_1,\omega_3,\omega_4\}$ und $|S| = \{(1,2), (1,3), (2,\omega_1)\}$.

Die Anfrage

$$<x/\tau_1 \mid (\exists y/\tau_2)(S(1,y) \wedge \neg S(x,y))>$$

hat $\{2\}$ als Antwort, wie sich leicht nachprüfen läßt (s.u.). Wenden wir die Umformungsregel 4 an, dann erhalten wir

$$\Pi<x/\tau_1,y/\tau_2 \mid S(1,y) \wedge \neg S(x,y)>.$$

Die Anfrage

$$<x/\tau_1,y/\tau_2 \mid S(1,y) \wedge \neg S(x,y)>$$

hat eine leere Antwort, da $\neg S(c_1,c_2)$ für kein Tupel $(c_1,c_2) \in \tau_1 \times \tau_2$ aus der Theorie ableitbar ist. Damit verlieren wir bei der Umformung gesicherte Information. Ableitbar ist dagegen $S(1,2) \wedge \neg S(2,2) \vee S(1,3) \wedge \neg S(2,3)$ wegen $\neg E(\omega_1,2) \vee \neg E(\omega_1,3)$, weshalb die ursprüngliche Anfrage eine nichtleere Antwort hat.

In beiden Fällen handelt es sich im wesentlichen um das gleiche Problem: Es gibt zu einem Tupel mit Konstanten aus Ω eine in der gegebenen Theorie ableitbare Disjunktion, die minimal ist. Eine Disjunktion $\Phi_1 \vee \Phi_2 \vee \ldots \vee \Phi_m$, $m > 1$, heißt *minimal* bezüglich einer Theorie $\mathcal{T}$, falls $\Phi_1 \vee \ldots \vee \Phi_{i-1} \vee \Phi_{i+1} \vee \ldots \vee \Phi_m$ für kein $i \in \{1,\ldots,m\}$ in $\mathcal{T}$ ableitbar ist. Diese Disjunktionen als Antworttupel betrachtet heißen auch *indefinite Antworttupel* ([Rei80]). Im Zusammenhang mit deduktiven Datenbanken werden sie als minimale ableitbare Grundformeln bezeichnet, wenn die einzelnen Φ_i atomare Formeln sind ([YH85]).

In modelltheoretischer Sichtweise bedeutet diese Minimalität für eine Belegung **t** der Variablen **x** im ersten Fall von oben: Die Disjunktion $\Phi_1(\mathbf{t}) \vee \Phi_2(\mathbf{t})$ ist in allen Modellen gültig, aber für kein $\Phi_i(\mathbf{t})$, $i = 1,2$, liegt in jedem Modell Gültigkeit vor. Dabei ergeben sich die Modelle durch Identifikation der Konstanten aus Ω mit Konstanten aus C. Dies ist nichts anderes als das Tautologieproblem, das wir unter anderem beim Zusammenfassen von Tupeln in Ergebnissen der "lazy evaluation" betrachtet haben (vergleiche Beispiel 6.7) und das Grund dafür ist, daß die V-Auswertung im allgemeinen unvollständige Ergebnisse liefert.

Bei der Überführung eines Existenz-quantifizierten Ausdrucks tritt genau das gleiche Problem auf, da hier nicht mehr für eine Belegung von **x** die Disjunktion mit allen Belegungen von y überprüft wird, sondern jede Kombination derartiger Belegungen einzeln.

Der Verzicht auf Vollständigkeit in Teilausdrücken zu Gunsten der Effizienz kann in Algebraausdrücken zu inkorrekten Antworten führen (betrachte als Beispiel $R \setminus (R \setminus S)$ und setze $R = \{\omega_1\}$ und $S = \{\omega_2\}$). In dem Verfahren von Reiter wird dieses Problem durch das Heranbringen aller Negationen an die atomaren Formeln und durch den Verzicht auf die Verwendung der Differenz als Operation vermieden. Der Verzicht auf die Differenz ist möglich, da die Bedeutung negierter atomarer Formeln

direkt definiert wird (die Differenz ist außerdem mit Hilfe von Kartesischem Produkt und Division darstellbar).

Die Vorgehensweise bei diesem Verfahren kann in unserer Betrachtungsweise wie folgt bei der Berechnung der uni-V-Antwort nachgebildet werden:

- Negationen in der Anfrage werden durch geeignete äquivalenzerhaltende Umformung an die Vergleichsausdrücke "herangeführt".

Sei $\mathfrak{W}$ die C-Relation, die durch anschließende "lazy evaluation" erhalten wird.

- Die Tupel aus $\mathfrak{W}$ werden bei der Bildung der Antwort alle einzeln betrachtet.
- Ein Tupel aus $\mathfrak{W}$ qualifiziert sich für die Antwort, falls sein Wert im COND-Attribut gemäß der Kleene'schen dreiwertigen Logik den Wert *wahr* hat, d.h. eine Tautologie darstellt. Dabei wird einem Vergleichsausdruck der Wert *unbekannt* zugeordnet, wenn in ihm genau eine Variable aus V vorkommt oder die beiden Operanden unterschiedliche Variable aus V sind.

Die Äquivalenz dieses Vorgehens mit der Reiter'schen Methode ergibt sich auf der Basis der Äquivalenz von modelltheoretischer und beweistheoretischer Sichtweise im Fall totaler DB-Zustände aus folgenden Feststellungen:

- Belegungstupel mit V-Variablen erfüllen ein Bereichsprädikat nur dann, wenn sie in der entsprechenden Relation enthalten sind. Dies entspricht der Auswertung einer primitiven Anfrage, für deren Ergebnistupel ebenfalls (syntaktisches) Enthaltensein erforderlich ist. Ist das Bereichsprädikat negiert, dann qualifizieren sich nur solche Belegungstupel, zu denen es keine verträglichen Tupel in der Relation gibt. Dies entspricht der Auswertung einer primitiven Anfrage mit negierter atomarer Formel.
- Ein universeller Quantor wird bei der "lazy evaluation" durch eine Konjunktion seiner Matrix ersetzt. Dabei werden für die gebundene Variable alle Belegungen eingesetzt, die sich gemäß dem zugehörigen Bereichsausdruck für die Variable ergeben. Dies entspricht genau dem Effekt, der durch die verwendete Form der Division in der Transformationsregel 3 bewirkt wird.
- Ein Existenz-Quantor wird bei der "lazy evaluation" durch eine Disjunktion ersetzt, wobei auch hier für die quantifizierte Variable alle Belegungen durchlaufen werden, die sich aus dem zugehörigen Bereichsausdruck ergeben. Durch die Projektion in der Transformationsregel 4 wird bewirkt, daß für eine beliebige Einsetzung eines Tupels **t** für die Variable **x** genau alle Werte des zugehörigen einfachen Typs Θ der quantifizierten Variablen zusammen mit **t** in Φ eingesetzt werden und die Menge der erhaltenen Ausdrücke als Disjunktion ausgewertet wird.
- Die Regeln 1 und 2 schließlich bewirken, daß eine Auflösung einer Anfrage in einen Ausdruck mit den Junktoren $\vee$, $\wedge$ und $\neg$ sowie den Operationen Division und Projektion erfolgt. Division und Projektion wirken bei Einsetzung eines Tupels für die freien Variablen wie Konjunktionen bzw. Disjunktionen. Daher erhalten wir mit der Einsetzung eines Tupels $(c_1,....,c_n)$ von Konstanten für die freien Variablen $(x_1,....,x_n)$ einer Anfrage $<(x_1/\tau_1,....,x_n/\tau_n) \mid \Phi(x_1,....,x_n)>$ durch die Transformationsschritte einen Ausdruck, in dem genau dann $(c_1,....,c_n)$ enthalten ist, wenn sich $(c_1,....,c_n)$ bei der Bildung von $\mathfrak{W}$ wie beschrieben qualifiziert.

Wir können damit feststellen, daß das Verfahren von Reiter zur gleichen gesicherten Antwort führt wie die "lazy evaluation", wenn die anschließende Auswertung

von Ausdrücken im COND-Attribut effizient mit Hilfe der dreiwertigen Logik erfolgt. Diese Vorgehensweise entspricht aber wiederum genau unserer V-Auswertung.

In [YC88] wird das Verfahren von Reiter aufgegriffen und ein Vorschlag gemacht, wie die Unvollständigkeit des Verfahrens überwunden werden kann. Dazu werden Relationen eingeführt mit Mengen von Tupeln als Elemente statt einzelner Tupel. Diese Mengen werden als Disjunktionen von Tupeln interpretiert, d.h. statt einer atomaren Formel $P(t_1,....,t_n)$ ist für ein Prädikat nur eine Disjunktion $P(t_{11},....,t_{1m_1})$ $\vee$ .... $\vee$ $P(t_{n1},....,t_{nm_n})$ aus der gegebenen Theorie ableitbar. Mit Hilfe dieser speziellen Tupel werden indefinite Antworten dargestellt, die schon in [Rei78] und [Rei80] als Antwortform betrachtet wurden. Bei der Zerlegungsregel für die Disjunktion werden indefinite Tupel mit in das Ergebnis aufgenommen. Diese Tupel entsprechen den $\omega_{\{\}}$-Tupeln, die wir in Kapitel 4 betrachtet haben. Da in diesem Vorschlag wegen der angestrebten Vollständigkeit alle ableitbaren indefiniten Tupel, die nicht von informativeren Tupeln impliziert werden, betrachtet werden, ist das Verfahren nicht effizient.

6.6 Bemerkungen zu einem intuitionistischen Ansatz

Die Effizienzprobleme bei der Bestimmung korrekter und vollständiger Antworten sind im wesentlichen darauf zurückzuführen, daß es (vermutlich) keine effiziente Lösung für das Tautologieproblem bei Booleschen Ausdrücken gibt. Eine naheliegende Idee ist es daher, für die Auswertung von Anfragen eine Logik zu verwenden, bei der Tautologien, die auf die Existenz von Nullwerten im DB-Zustand zurückzuführen sind, nicht auftreten. Interpretationen von Antworten sind dann gemäß der verwendeten Logik anzugeben. Dabei sollte die Definition der Korrektheit zur bisher verwendeten Definition passen, um zu gewährleisten, daß in Antworten nur gesicherte Information enthalten ist. Vollständigkeit muß in der neuen Logik natürlich schwächer definiert sein.

Eine in dieser Hinsicht geeignete Logik ist die von L.E.J. Brouwer begründete *intuitionistische Logik* (siehe hierzu etwa [Fit69]). Da in dieser Logik nicht der Satz vom ausgeschlossenen Dritten (*tertium non datur* Prinzip) gilt, ist z.B. $a \vee \neg a$, a Aussagenvariable, keine Tautologie wie in der zweiwertigen Booleschen Logik. Die Umformungsregeln im Verfahren von Reiter passen zu dieser Eigenschaft der intuitionistischen Logik (betrachte die Inklusionen in den Regeln 2 und 4, die zur Unvollständigkeit des Verfahrens führen). In [Lak89] wird dies aufgegriffen und der Versuch unternommen, auf der Grundlage der intuitionistischen Logik Auswertungsverfahren anzugeben, die vollständig und korrekt hinsichtlich passend gewählter Definitionen sind.

Wir zeigen im folgenden, daß die in [Lak89] behauptete Äquivalenz von intuitionistischer Relationenalgebra und logischer Anfragesprache nicht gilt. Die angegebene Relationenalgebra eignet sich nur dazu, logische Anfragen in der von Reiter angegebenen Weise zu zerlegen und dabei die Inklusionen durch Gleichungen zu ersetzen.

Wir benötigen zunächst einige Definitionen. Dabei gehen wir davon aus, daß wieder eine relationale Sprache $\mathcal{R} = (\mathcal{A},\mathcal{W})$ gegeben ist wie in Abschnitt 6.5 und eine erweiterte relationale Theorie $\mathcal{T}$ zu $\mathcal{R}$ einen Datenbankzustand festlegt.

Zwei Tupel s und t (eventuell mit Elementen aus Ω) sind *verschieden* bezüglich einer Theorie $\mathcal{T} \Leftrightarrow_{df} \mathcal{T} \vdash_i s \neq t$, wobei $\vdash_i$ intuitionistische Beweisbarkeit bedeutet.

Bemerkung: Wir benötigen hier kein genaues Verständnis von intuitionistischer Beweisbarkeit und verweisen daher auf die Literatur (etwa [Fit69]).

Aussagen können unterschieden werden in solche, die *mit Sicherheit* gültig sind und solche, deren Gültigkeit *unbestimmt* ist. Dazu werden die Aussagen mit T bzw. I gekennzeichnet (in [Lak89] werden eine Variante der "model sets" von Hintikka und der "signed formulas" von Smullyan [Smu68] verwendet, bei der mit F (für *falsch*) gekennzeichnete Aussagen sich über die CWA ergeben).

Zwei Tupel s und t heißen *mit Sicherheit verschieden* $\Leftrightarrow_{df} \mathcal{T} \vdash_i T\ s \neq t$.
Zwei Tupel s und t heißen *unbestimmt verschieden* $\Leftrightarrow_{df} \mathcal{T} \vdash_i I\ s \neq t$.

Eine äquivalente Definition für *unbestimmt verschieden* kann unter Bezug auf die Eigenschaft *mit Sicherheit verschieden* gegeben werden ([Lak89]):

Zwei Tupel s und t sind *unbestimmt verschieden* $\Leftrightarrow_{df}$
- s ist nicht mit Sicherheit verschieden von t und
- s ist nicht syntaktisch gleich mit t.

Ähnlich wie in den von Biskup eingeführten erweiterten Relationen, können auch in V-Relationen Tupel unterschieden werden in solche, die mit Sicherheit und solche, die nur unbestimmt in einer Relation enthalten sind. Zu dieser Unterscheidung werden in [Lak89] die beiden Funktionen *certain* und *indefinite* (kurz c und i) benutzt.

Betrachten wir die Definition der Differenz, wie sie mit diesen Begriffen in [Lak89] gegeben wird:

Seien R und U V-Relationen über der gleichen Attributmenge;

$R \setminus U =_{df} T$ mit

$c(T) = \{r \in c(R) \mid r$ ist mit Sicherheit verschieden von jedem $u \in U\}$

$i(T) = \{r \in R \mid r$ ist unbestimmt verschieden von einem $u \in U$ und entweder mit Sicherheit oder unbestimmt verschieden von allen anderen Tupeln von T$\}$

Seien die drei folgenden V-Relationen über der gleichen Attributmenge gegeben:

$R = \{(1,\omega_1)\}$, $U = \{(1,\omega_2)\}$ und $W = \{(1,\omega_3)\}$.

Für den Ausdruck

$R \setminus ((R \setminus U) \setminus (R \setminus W))$

ergibt sich dann folgende Auswertung (wir kennzeichnen die Tupel mit c und i gemäß ihrer Zuordnung nach Definition der Differenz):

$R \setminus U$: $\{(1,\omega_1,c)\} \setminus \{(1,\omega_2,c)\} = \{(1,\omega_1,i)\}$

$R \setminus W$: $\{(1,\omega_1,c)\} \setminus \{(1,\omega_3,c)\} = \{(1,\omega_1,i)\}$

$(R \setminus U) \setminus (R \setminus W)$: $\{(1,\omega_1,i)\} \setminus \{(1,\omega_1,i)\} = \emptyset$

$R \setminus ((R \setminus U) \setminus (R \setminus W))$: $\{(1,\omega_1,c)\} \setminus \emptyset = \{(1,\omega_1,c)\}$

Wir erhalten ein Ergebnis $(\{(1,\omega_1,c)\})$, das keine gesicherte Information darstellt: Werden ω_2 und ω_3 mit verschiedenen Werten und ω_1 und ω_2 mit dem gleichen Wert belegt, dann liefert der Ausdruck eine leere Relation als Ergebnis.

Mit der von uns in Abschnitt 6.1 angegebenen switch-Auswertung ergibt sich zum Vergleich folgende Auswertung:

$R \setminus ((R \setminus U) \setminus (R \setminus W)) \rightarrow R \setminus_s ((R \setminus_m U) \setminus_m (R \setminus_s W))$

$R \setminus_m U$: $\{(1,\omega_1)\} \setminus_m \{(1,\omega_2)\} = \{(1,\omega_1)\}$

$R \setminus_s W$: $\{(1,\omega_1)\} \setminus_s \{(1,\omega_3)\} = \Phi$

$(R \setminus_m U) \setminus_m (R \setminus_s W)$: $\{(1,\omega_1)\} \setminus_m \Phi = \{(1,\omega_1)\}$

$R \setminus_s ((R \setminus_m U) \setminus_m (R \setminus_s W))$: $\{(1,\omega_1)\} \setminus_s \{(1,\omega_1)\} = \Phi$

Durch den intuitionistischen Ansatz sollte nicht die Korrektheit von Ergebnissen neu definiert werden, sondern nur für die Vollständigkeit eine etwas schwächere Form verlangt werden. Das Beispiel zeigt, daß mit der gewählten Definition für die Differenz dieses Ziel nicht erreicht werden kann. Die in [Lak89] angegebene intuitionistische Relationenalgebra eignet sich nur als Zielsprache für die Übersetzung der betrachteten logischen Sprache. Die behauptete Äquivalenz gilt nicht. Um sie zu erreichen, kann ähnlich wie bei der switch-Auswertung vorgegangen werden. Wir gehen hierauf aber nicht weiter ein.

7 Vollständig fehlende Information zu Attributwerten

7.1 Problemstellung

Bei unseren bisherigen Betrachtungen sind wir davon ausgegangen, daß mit einem fehlenden Attributwert das Wissen um die Existenz eines oder mehrerer Werte an der betreffenden Stelle verknüpft ist. Fehlt auch dieses Wissen, d.h. ist zu einem fehlenden Attributwert keine Information vorhanden ("nichts bekannt", "no information"), dann kommt ein möglicher Fall hinzu, der bei der Interpretation von partiellen Tupeln, Relationen und Zuständen zu beachten ist:

Ein fehlender Attributwert in einem Tupel kann bedeuten, daß auch für den modellierten Gegenstand kein entsprechender Wert vorhanden ist.

Es kann viele Gründe geben, warum für einen Gegenstand kein Wert vorhanden ist. Mögliche Gründe sind zum Beispiel, daß ein Wert geheimgehalten oder wegen Ungenauigkeit nicht freigegeben wird. Wir wollen hier derartige Interpretationsmöglichkeiten nicht betrachten, sondern annehmen, daß ein fehlender Wert für einen Gegenstand bedeutet, daß es diesen Wert definitiv nicht gibt, unser Wissen über den zu modellierenden Sachverhalt also vollständig ist.

Unter dieser Annahme können zwei Gründe für das Fehlen von Werten unterschieden werden:

- Ein Wert kann fehlen, da eine Eigenschaft einem Objekt nicht zugeordnet werden kann (z. B.: der Geburtsname einer ledigen Person). Für diesen Fall paßt die in [LeLi86] gewählte Charakterisierung "non-applicable". Die Zuordnung einer Eigenschaft kann eventuell erfolgen, wenn sich andere Eigenschaften des Objekts, d.h. das Objekt in seiner beobachteten Erscheinung, geändert haben.
- Ein Wert kann fehlen, da eine Beziehung zwischen Objekten nicht vorhanden ist und Beziehungen zusammen mit anderen Gegenständen modelliert werden. Betrachte zum Beispiel einen Relationstyp **ANGEBOT** (ARTNR, BEZEICHNUNG, PREIS, LNR), durch den Angaben zu Artikeln einschließlich ihrer Beziehungen zu dem jeweiligen Lieferanten modelliert werden. Wird ein Artikel von keinem Lieferanten geliefert, fehlt ein Wert im Attribut LNR.

Wenn wir die Festlegung der Bedeutung eines partiellen Zustands (im folgenden auch *Basiszustand* genannt) mit der "nichts bekannt"-Interpretation fehlender Attributwerte wie bei Zuständen mit unbekannten Attributwerten über die Menge möglicher Zustände vornehmen, ergeben sich zwei Formen möglicher Zustände:

- totale Zustände, wie wir sie bisher ausschließlich betrachtet haben, und
- Zustände, in denen Relationen mit partiellen Tupeln vorkommen, wobei fehlende Attributwerte in diesen Tupeln zu interpretieren sind als: *Wert nicht vorhanden* ("non-existent"). Mit dieser Interpretation modellieren partielle Tupel vollständige Information.

Mögliche Relationen zu einer partiellen Relation können damit wie folgt erhalten werden:

Fehlende Attributwerte werden entweder durch einen oder mehrere beliebige konkrete Werte des zugehörigen Wertebereichs oder durch ein spezielles Symbol

ersetzt. Jedes Vorkommen des gewählten speziellen Symbols wird mit der Bedeutung versehen, daß auch für den entsprechenden modellierten Fakt an der betreffenden Stelle kein Wert vorhanden ist.

Mit dieser Form der Semantikdefinition ergibt sich das Problem, daß die Bedeutung von Basiszuständen nicht mehr wie bisher direkt mit Hilfe der Bedeutung gewöhnlicher, d.h. totaler Zustände festgelegt werden kann. Welche Probleme auftreten können, wollen wir im folgenden mit Hilfe von Beispielen untersuchen.

Sei ein Relationstyp **GAST** (NAME, ORT, BEGLEITUNG) gegeben. Durch ihn soll Information über Gäste einer privaten Feier modelliert werden. Von jedem Gast soll neben seinem Namen und dem Wohnort vermerkt werden, in welcher Begleitung er oder sie kommt. NAME sei Schlüssel für den Relationstyp. Die Werte in den Attributen NAME und ORT sollen stets definiert sein; in BEGLEITUNG dagegen soll ein konkreter Wert, ein Name, fehlen können mit der Bedeutung, daß wir keine Information darüber haben, ob ein Gast in Begleitung kommt oder nicht. Ein fehlender Wert in diesem Attribut kann demnach bedeuten, daß ein Gast in Begleitung kommt, der Name der Begleitung aber nicht bekannt ist, oder daß der Gast allein kommt.

Sei folgende Beispielrelation zu dem Relationstyp **GAST** gegeben:

GAST

NAME	ORT	BEGLEITUNG
Harald	Toulouse	Elvira
Gisela	Bonn	-

Da wir bezüglich der Begleitung von Gisela nichts wissen, ergeben sich für diese partielle Relation als mögliche Relationen alle totalen Relationen, in denen ein Wert aus dem Wertebereich von BEGLEITUNG für "-" eingetragen ist (d.h. alle Vervollständigungen der Relation), sowie die Relation selbst mit einer in "Wert nicht vorhanden" geänderten Bedeutung von "-" bzw. eines anderen speziellen Symbols, das für "-" eingesetzt wird.

Betrachten wir die zuletzt genannte mögliche Relation. Derartige Relationen werden bei der Einführung von Normalformen gerne als Beispiel genommen, um sogenannte "Einfüge-Anomalien" zu demonstrieren. Einfüge-Anomalien können auftreten, wenn unterschiedliche *Konzepte* in einem Relationstyp modelliert werden. Einige dieser Anomalien lassen sich mit funktionalen oder mehrwertigen Abhängigkeiten formal fassen und durch Übergang zu Zerlegungen mit Relationstypen in höheren Normalformen vermeiden. In unserem Beispiel ist der Relationstyp allerdings schon in vierter Normalform (alle mehrwertigen Abhängigkeiten sind trivial). Trotzdem finden sich zwei Konzepte, die durch ihn modelliert werden: Es werden Name und Wohnort der Gäste erfaßt, wodurch festgelegt wird, wer ein Gast ist; zusätzlich wird zu jedem Gast der Name der begleitenden Person - falls vorhanden - gespeichert. Demnach sind andere formale Hilfsmittel als funktionale und mehrwertige Abhängigkeiten notwendig, um geeignete Zerlegungen finden zu können.

Für eine Modellierung des Sachverhalts im Entity/Relationship-Datenmodell bietet sich ein Schema an mit zwei Entity-Typen,

GAST (mit den Attributen NAME und ORT) und

BEGLEITUNG (mit dem Attribut NAME),

sowie einem Relationship-Typ, der beide Entity-Typen verknüpft.

Damit ergibt sich im Fall eines Gastes ohne Begleitung keinerlei Problem, da in einem entsprechenden Zustand des Entity/Relationship-Schemas dieser Fall durch das Fehlen einer Beziehung für den Gast repräsentiert werden kann. Die beiden Konzepte, ein Gast mit Angaben zu seinem Namen und Wohnort sowie die Begleitung, in der er oder sie kommt, sind durch verschiedene Schemaelemente modelliert.

Es ist naheliegend, für das relationale Schema zur Vermeidung der Anomalie eine dem Entity/Relationship-Schema entsprechende Modellierungsform zu wählen, etwa ein Schema mit den Relationstypen

GAST (NAME, ORT) und

BEGLEITUNG (NAME, B_NAME).

Mit diesem Schema kann die Information, daß ein Gast ohne Begleitung kommt, dadurch repräsentiert werden, daß ein Tupel mit dem Namen dieses Gastes zwar in der Relation **GAST**, nicht aber in der Relation **BEGLEITUNG** vorkommt. Dies ist möglich, da für die Interpretation von Zuständen sowie bei der Auswertung von Anfragen die CWA zugrunde gelegt wird.

Gibt es für jeden Relationstyp einen Schlüssel ohne Attribute, in denen fehlende Werte auftreten können, dann lassen sich für die Bedeutung "kein Wert vorhanden" Zerlegungen angeben, in denen alle Relationen total sind und unter denen die mit dem Ausgangsschema modellierbare Information ohne Verlust darstellbar bleibt. Dies folgt direkt aus einem Satz der Zerlegungstheorie ([Ris77]). Damit könnte die Bedeutung von Zuständen mit fehlenden Attributwerten unter der genannten Bedingung auf die Bedeutung von totalen Zuständen zurückgeführt werden. Es ist allerdings zu beachten, daß die Anzahl der notwendigen Relationstypen in einer Zerlegung exponentiell sein kann in der Anzahl der Attribute, in denen fehlende Werte erwartet werden. Wie wir oben schon gesehen haben, benötigen wir auch Mittel, um Konzepte beim Zerlegen geeignet zu berücksichtigen.

Betrachten wir nun wieder die Bedeutung "nichts bekannt" im Fall fehlender Attributwerte. Für unser Beispiel ergibt sich das Problem, in einem Zustand die Information, daß nichts über die Begleitung eines Gastes bekannt ist, unterzubringen. Fehlt für einen solchen Gast ein Tupel in der Relation **BEGLEITUNG**, dann führt die CWA dazu, daß dieses Fehlen interpretiert wird als: Der Gast kommt mit Sicherheit ohne Begleitung. Diese Interpretation ist aber nur zutreffend für Tupel in möglichen Zuständen, in denen im Attribut B_NAME ein Wert fehlt.

Sei das DB-Schema von oben mit den beiden Relationstypen **GAST** (NAME, ORT) und **BEGLEITUNG** (NAME, B_NAME) gegeben. Betrachte folgenden Zustand zu diesem Schema, in dem gegenüber dem Zustand von oben Daten zu Frank als weiterem Gast gespeichert sind:

GAST

NAME	ORT
Harald	Toulouse
Gisela	Bonn
Frank	Kiel

BEGLEITUNG

NAME	B_NAME
Harald	Elvira
Gisela	-

Folgende Semantik ergibt sich mit der CWA und der Bedeutung "nichts bekannt" von "-":

Für Gisela ist nicht bekannt, ob sie in Begleitung kommt oder nicht;

von Frank wissen wir, daß er ohne Begleitung kommt.

Betrachte nun den möglichen Zustand, der sich dadurch ergibt, daß "-" die Bedeutung "kein Wert vorhanden" zugeordnet wird. In diesem möglichen Zustand ist das modellierte Wissen zu der Begleitung von Frank und Gisela gleich: Beide kommen mit Sicherheit ohne Begleitung. Die Modellierung dieses Wissens erfolgt aber unterschiedlich: Für Frank ergibt es sich mit der CWA durch das Fehlen eines Tupels mit Frank als Wert im Attribut NAME in der Relation **BEGLEITUNG**; für Gisela gibt es ein Tupel in der Relation **BEGLEITUNG**, das Fehlen einer Begleitung ergibt sich aus dem Wert "-" im Attribut B_NAME. Eine Möglichkeit, dieses Problem zu vermeiden, besteht in der Abänderung der Definition möglicher Zustände gemäß der oben diskutierten Anomalievermeidung: Es werden bei der Bildung möglicher Zustände alle Tupel aus der Relation **BEGLEITUNG** gestrichen, für die in B_NAME ein undefinierter Wert fehlt.

Mit der unterschiedlichen Darstellung des gleichen Sachverhalts für Gisela und Frank gemäß der ursprünglichen Definition möglicher Zustände kann es zu verschiedener Berücksichtigung gleichwertiger Information bei der Auswertung von Anfragen und damit zu Problemen mit der Interpretation von Ergebnissen kommen. Betrachte folgende Anfrage:

"*Gib die Daten aller Gäste, die ohne Begleitung kommen*".

Eine naheliegende Formulierung dieser Anfrage im TRC lautet:

$(x)/\ \text{GAST}\ x \wedge \neg (\exists y\text{: BEGLEITUNG}\ y)(y.\text{NAME} = x.\text{NAME})$

Da kein Bezug auf ein Attribut mit fehlendem Wert genommen wird, kann die Anfrage wie im Fall totaler Zustände ausgewertet werden. Die einzig notwendige Erweiterung besteht darin, Variable mit partiellen Tupeln belegen zu können. Als Antwort erhalten wir dann {(Frank, Kiel)}. Das Tupel (Gisela, -) fehlt in der Antwort, obwohl für diesen (möglichen) Zustand als bekannt angenommen ist, daß Gisela ohne Begleitung kommt.

Es stellt sich die Frage, unter welchen Bedingungen die Lösung des Problems für dieses Beispiel, das Weglassen partieller Tupel aus möglichen Relationen, angewendet werden kann. Die Bemerkungen im Zusammenhang mit der Einfüge-Anomalie haben deutlich gemacht, daß dazu Konzepte von Bedeutung sind.

Betrachten wir ein weiteres Beispiel mit einem DB-Schema, das aus den beiden Relationstypen

ANGESTELLTE (NAME, ORT) und

PROJEKTLEITUNG (NAME, PROJNR)

besteht. Sei folgender Sachverhalt durch das Schema modelliert:

Angestellte können als Projektleiter ein oder mehrere Projekte leiten; es ist aber auch möglich, daß ein Angestellter zum Projektleiter ernannt wird, ohne daß er die Leitung eines Projektes übernimmt, oder daß ihm nach Abschluß eines Projektes nicht sofort die Leitung eines neuen Projektes übertragen wird.

Bemerkung: Das Schema ist vielleicht kein gelungener Entwurf für die Modellierung des beschriebenen Sachverhalts. Es mag uns hier aber als einfaches Beispiel für ein allgemeineres Problem dienen, das häufig in Anwendungen zu finden ist, wenn bewußt auf höhere Normalformen verzichtet wird (siehe hierzu etwa die Argumentation in [KS95]).

In einem erweiterten ER-Modell oder einem objektorientierten Datenmodell könnte der beschriebene Sachverhalt zum Beispiel wie folgt modelliert werden:
Es gibt drei Entity-(Objekt-) Typen, und zwar:

ANGESTELLTE mit den Attributen NAME und ORT,

PROJEKTLEITUNG (ohne Attribut) und

PROJEKT mit dem Attribut PROJNR und eventuell weiteren Attributen.

PROJEKTLEITUNG wird zum Subtyp von **ANGESTELLTE** bestimmt und über einen binären Beziehungstyp mit **PROJEKT** verknüpft.

Im angegebenen relationalen Schema wird durch das Auftreten eines Namens im Attribut NAME des Relationstyps **PROJEKTLEITUNG** eine Eigenschaft festgelegt (Projektleiter zu sein) unabhängig davon, ob im Attribut PROJNR ein konkreter Wert steht oder nicht. Wissen wir nichts über einen Wert im Attribut PROJNR, dann kann unabhängig davon Wissen bezüglich der Eigenschaft, Projektleiter zu sein, vorliegen. Es sind zwei Konzepte in einem Relationstyp modelliert. Die folgende Anfrage spricht nur das Konzept "Angestellter ist Projektleiter" an:

"*Gib die Daten aller Angestellten, die nicht Projektleiter sind*".

Die Formulierung

$$(x)/\ \text{ANGESTELLTE}\ x \wedge \neg(\exists y{:}\ \text{PROJEKTLEITER}\ y)(y.\text{NAME} = x.\text{NAME})$$

liefert in diesem Fall für alle möglichen Zustände, die ohne Streichung partieller Tupel erhalten werden, genau die gewünschte Antwort. Man beachte den Unterschied zur Anfrage von oben, in der nach den Daten derjenigen Gäste gefragt wird, die ohne Begleitung kommen.

Für die Anfrage

"*Gib die Daten aller Angestellten, die kein Projekt leiten*"

stellt die angegebene TRC-Anfrage dagegen eine Formulierung dar, die im allgemeinen zu unvollständigen Antworten führt, wenn partielle Tupel nicht aus möglichen Zuständen entfernt werden. Es fehlen die Daten derjenigen Angestellten, die zwar Projektleiter sind, aber kein konkretes Projekt leiten.

Durch das Entfernen von Tupeln mit fehlendem definiertem Wert im Attribut PROJNR geht die Information verloren, daß der betreffende Angestellte Projektleiter ist. Bezieht sich eine Anfrage auf das Konzept Projektleiter, dann kann

durch diesen Verlust eine Antwort erhalten werden, die unkorrekte Information darstellt. Dies zeigt, daß ein Streichen von partiellen Tupeln in möglichen Zuständen im allgemeinen nicht ohne Beachtung von Konzepten erfolgen sollte. Es muß daher eine weitere Zerlegung von Relationstypen gemäß vorhandener Konzepte vorgenommen bzw. ein anderer Schemaentwurf gemacht werden, oder partielle Tupel müssen in möglichen Zuständen erlaubt sein. Da eine Zerlegung oder ein anderes Schema ohne das genannte Problem wegen einer zu großen Anzahl notwendiger Relationstypen nicht immer sinnvoll ist, wollen wir partielle Tupel in Zuständen zulassen und nach einem Weg suchen, den Begriff "Konzept" zu präzisieren und Konzepte bei der Definition möglicher Zustände geeignet zu berücksichtigen.

Betrachten wir zunächst ein weiteres Beispiel mit einem Relationstyp, durch den zwei Konzepte modelliert werden. Sei das **GAST/BEGLEITUNG**-Beispiel wie folgt erweitert:

Zusätzlich zum Namen einer eventuell den Gast begleitenden Person soll vermerkt werden, welchen Beitrag der Gast zum kalten Buffet leisten will. Dazu fügen wir dem Relationstyp **BEGLEITUNG** das Attribut BEITRAG hinzu und benennen den neuen Relationstyp in **BEGL_BUFFET** um. {NAME} soll weiterhin Schlüssel sein. Folgender Zustand sei gegeben:

GAST

NAME	ORT
Harald	Toulouse
Gisela	Bonn
Frank	Kiel
Marga	Trier

BEGL_BUFFET

NAME	B_NAME	BEITRAG
Harald	Elvira	-
Gisela	-	Salat
Marga	-	-

Im Unterschied zum Relationstyp **PROJEKTLEITUNG** von oben mit den Attributmengen {NAME} und {NAME, PROJNR} zu den beiden modellierten Konzepten "Angestellter x ist Projektleiter" und "x leitet Projekt y" besteht in diesem Beispiel keine Teilmengenbeziehung zwischen den Attributmengen {NAME, B_NAME} und {NAME, BEITRAG} der in **BEGL_BUFFET** modellierten Konzepte "x kommt in Begleitung von y" und "x steuert z zum Buffet bei". Außerdem ist durch den Schlüssel {NAME} nicht die Attributmenge eines Konzeptes gegeben.

Es kann sich also wie für das Schema mit dem Relationstyp **BEGLEITUNG** der Fall ergeben, daß in einer Anfrage zwar ein Attribut des Relationstyps **BEGL_BUFFET**, nicht aber eines der beiden modellierten Konzepte angesprochen wird. Zusätzlich kann wie beim Relationstyp **PROJEKTLEITUNG** der Fall eintreten, daß auf nur eines von zwei modellierten Konzepten Bezug genommen wird.

Betrachten wir etwas genauer Einfügeoperationen für unsere Beispielschemata {**GAST**(....), **BEGLEITUNG**(....)} und {**GAST**(....), **BEGL_BUFFET**(....)}. Ist beim Einfügen der Daten eines Gastes bekannt, daß er ohne Begleitung kommt (erstes Schema) bzw. ohne Begleitung kommt und auch nichts mitbringt (zweites Schema), dann erfolgt keine Eintragung in den Relationen **BEGLEITUNG** bzw. **BEGL_BUFFET**. Dies ist in den Beispielzuständen der Fall für den Gast Frank. Die Darstellung von Information in einem möglichen Zustand sollte mit ihrer Darstellung im

Basiszustand übereinstimmen. Daraus ergibt sich, daß die Tupel (Gisela, -) und (Marga, - , -) in den zum ersten bzw. zweiten Schema angegebenen Zuständen als nicht existent angesehen werden müssen, wenn diese Zustände als mögliche Zustände mit entsprechend geänderter Bedeutung der fehlenden Werte betrachtet werden. Es gibt in diesem Fall keinen Grund, die Informationen zu Frank und Gisela bzw. Frank und Marga unterschiedlich zu repräsentieren. Die Eintragung der beiden Tupel im Basiszustand erfolgte nur deshalb, weil nicht ausgeschlossen werden kann, daß beide Gäste in Begleitung kommen oder (im zweiten Fall) etwas mitbringen.

Im zweiten Schema haben wir noch weitere (symmetrische) Fälle zu betrachten: Auch wenn für Gisela bekannt ist, daß sie ohne Begleitung kommt, muß das Tupel (Gisela, -, Salat) in **BEGL_BUFFET** eingetragen werden, um die Information, daß Gisela Salat mitbringt, unterbringen zu können. Analog verhält es sich mit dem Tupel (Harald, Elvira, -). Wir können demnach nicht, wie davor, eine einfache Streichung von partiellen Tupeln bei der Bildung möglicher Zustände vornehmen. Dies hat wie für das Beispielschema {(**ANGESTELLTE**(....), **PROJEKTLEITUNG**(....)} zur Folge, daß bei der Auswertung von Anfragen beachtet werden muß, welche Konzepte in der Anfrage angesprochen werden.

7.2 Semantik von DB-Zuständen

Die Beispiele zeigen, daß zwei Arten partieller Tupel bei der Bildung möglicher Zustände zu unterscheiden sind:

- Partielle Tupel, die aus einer Relation entfernt werden können, ohne daß sich die repräsentierte Information ändert. Durch das Entfernen wird *negative* Information, die explizit repräsentiert wurde, mit der CWA zu impliziter Information.
- Partielle Tupel, die bezüglich der darzustellenden Information nicht redundant sind und bei der Auswertung von Anfragen in Abhängigkeit von den angesprochenen Konzepten beachtet werden müssen.

Bevor wir Angaben zur Zuordnung von partiellen Tupeln zu einer dieser beiden Tupelarten machen können, müssen wir präzisieren, was unter einem Konzept verstanden werden soll.

Ein Konzept im relationalen Datenmodell stellt eine Aussagenform für Kombinationen von Werten einer festen Attributmenge eines Relationstyps dar. Dabei ist für jedes Attribut festgelegt, ob es zur Aussage immer einen konkreten Wert beisteuern muß (obligatorisches Attribut) oder ob der zugehörige Wert in der Aussage auch undefiniert (im Sinne von "nicht vorhanden") sein darf (optionales Attribut), ohne daß die Aussage ihren Sinn verliert.
Wann eine Aussage ihren Sinn verliert, ist Festlegungssache beim Modellieren und muß nicht weiter formalisiert werden.

In jedem Relationstyp soll ein Konzept durch die zugehörige Attributmenge eindeutig bestimmt sein. Falls dies nicht der Fall wäre, müßten Attributkombinationen eines Tupels in Abhängigkeit anderer Werte des Tupels interpretiert werden, was nicht sinnvoll erscheint. Wir benutzen daher Attributmengen zum Ansprechen von Konzepten.

Da wir davon ausgehen wollen, daß Relationstypen eines DB-Schemas stets in höherer Normalform sind (3. Normalform oder Boyce-Codd-Normalform) und

Konzepte zu Normalformen passen sollen, fordern wir von der Menge der Konzepte eines Relationstyps:

- Der Hypergraph, der zu einem Relationstyp erhalten wird, indem die Attribute als Knoten und die Konzepte als Kanten betrachtet werden, ist zusammenhängend.

Die Forderung besagt, daß Konzepte nicht "nebeneinander stehen" dürfen und alle Attribute eines Relationstyps erfassen müssen.

Es wird nicht gefordert, daß jedes Konzept eines Relationstyps den Primärschlüssel enthält. Eine solche Forderung wäre zu streng, wie der folgende Relationstyp zeigt, den wir aus dem Typ **PROJEKTLEITUNG** (NAME, PROJNR) erhalten, indem wir das Attribut ORT und folgende funktionalen Abhängigkeiten hinzufügen:

{PROJNR, ORT} $\rightarrow$ {NAME} und {NAME} $\rightarrow$ {ORT}.

Konzepte seien hier:

{NAME}: Angestellter x ist Projektleiter;

{PROJNR, ORT}: Projekt y wird in Ort z durchgeführt;

{NAME, ORT}: Angestellter x leitet (nur) Projekte in Ort z;

{PROJNR, NAME, ORT}: Angestellter x leitet Projekt y in Ort z.

Aus den Formulierungen ergibt sich, daß alle Attribute obligatorisch sind. Das Beispiel deutet an, daß auf die Unterteilung von Attributen in optionale und obligatorische Attribute verzichtet werden kann, wenn für alle entsprechenden Teilmengen Konzepte angegeben sind. Zusätzlich muß aber gewährleistet sein, daß alle Konzepte verträglich miteinander sind im Sinne der *Annahme der eindeutigen Beziehung zwischen Elementen von Attributmengen*, einer Annahme. die im Zusammenhang mit Universalrelationen-Schnittstellen gemacht wird ([MUV84]).

Betrachten wir die vorangehenden Beispiele. Wir haben oben das Konzept "Gast" (oder etwas ausführlicher: "x aus o ist Gast") benutzt, zu dem die Attribute NAME und ORT gehören. NAME muß obligatorisches Attribut sein, wenn das Konzept dem Sprachgebrauch von Gast entsprechen soll; ORT könnte als optionales Attribut festgelegt werden, wenn auch Gäste ohne festen Wohnsitz erwartet werden. Eine Aussage "Gast x kommt in Begleitung von y" macht nur einen Sinn, wenn zu x eine Begleitung existiert. Daher sollten für dieses Konzept die beiden Attribute NAME und B_NAME obligatorische Attribute sein.

Zum ursprünglichen Relationstyp **PROJEKTLEITUNG** ergeben sich mit der geschilderten äußeren Semantik die Konzepte "x ist Projektleiter" und "x leitet Projekt mit der Nummer y" mit den obligatorischen Attributen NAME bzw. NAME und PROJNR.

Zum Relationstyp **BEGL_BUFFET** (NAME, B_NAME, BEITRAG) gehören die beiden Konzepte "Gast x kommt in Begleitung von y" und "Gast x leistet den Beitrag z zum Buffet" mit den obligatorischen Attributen NAME und B_NAME bzw. NAME und BEITRAG. Ist in einem Tupel t für B_NAME oder BEITRAG kein Wert vorhanden, dann bedeutet dies, daß in t zu dem entsprechenden Konzept keine Information repräsentiert wird. Sind in B_NAME und BEITRAG keine Werte vorhanden, dann kann t aus dem Zustand entfernt werden, da zu den beiden einzigen in **BEGL_BUFFET** modellierten Konzepten keine Information in t enthalten ist.

Bemerkung: Im Zusammenhang mit Universalrelationen-Schnittstellen werden in [Mai83] ausführlich *Objekte* für das relationale Datenmodell diskutiert. Objekte entsprechen in etwa den hier betrachteten Konzepten. Es wird nur nicht zwischen obligatorischen und optionalen Attributen unterschieden, d.h. Objekte sind allein durch Angabe einer Attributmenge bestimmt.

Wir gehen im folgenden davon aus, daß zu den Relationstypen eines DB-Schemas stets Konzepte gehören. Falls keine Konzepte explizit angegeben sind, wird die gesamte Attributmenge des Relationstyps als Konzept angesehen, wobei alle Attribute obligatorisch sind. Mit diesen Voraussetzungen können wir nun mögliche Relationen und mögliche Zustände zu Basiszuständen mit fehlenden Attributwerten interpretiert als "nichts bekannt" angeben. Dabei schließen wir mit Hilfe der Konzepte bestimmte Zustände aus. Partielle Relationen und Zustände sollen wieder als ω-Relationen und ω-Zustände repräsentiert werden.

Die nachfolgende Definition charakterisiert diejenigen Relationen, in denen durch Tupel nur Aussageformen repräsentiert werden, die verträglich sind mit der gegebenen Konzeptmenge.

Definition: Sei RT ein Relationstyp mit Attributmenge α und Konzeptmenge $\mathfrak{K}$. Ein ω-Tupel t über α heißt erlaubt bzgl. $\mathfrak{K}$, falls gilt:

- t ist nicht das total undefinierte Tupel.
- Ist t in einem Attribut A definiert, dann gibt es mindestens ein Konzept aus $\mathfrak{K}$, in dem A enthalten ist und für das t in allen obligatorischen Attributen definiert ist.

Eine ω-Relation zu RT heißt erlaubt bzgl. $\mathfrak{K}$, falls sie nur erlaubte Tupel enthält.

Durch diese Definition wird gewährleistet, daß in erlaubten Relationen kein Tupel vorkommt, das einen definierten Wert enthält, der keiner sinnvollen Aussage zugeordnet werden kann.

In der folgenden Definition wird berücksichtigt, daß jeder fehlende Attributwert unabhängig von jedem anderen Wert zwei unterschiedliche Bedeutungen haben kann. Da wir angenommen haben, daß ein unbekannter Wert genau einen konkreten Wert repräsentiert (d.h. wir haben $POSS_C$ als Möglichkeitsfunktion gewählt), müssen wir Erweiterungen von ω-Relationen betrachten.

Definition: Sei R eine beliebige ω-Relation zu einem Relationstyp RT mit Konzeptmenge $\mathfrak{K}$. Eine ω-Relation R' heißt mögliche Relation zu R bzgl. $\mathfrak{K}$, falls R' aus einer Erweiterung R'' von R erhalten werden kann, indem aus R'' alle Tupel entfernt werden, die nicht erlaubt sind.
Sei σ ein DB-Schema, für das zu jedem seiner Relationstypen RT_i Konzepte angegeben sind, und sei z_ω ein ω-Zustand zu σ. Ein bezüglich der gegebenen Konzepte möglicher Zustand z zu z_ω wird aus z_ω erhalten, indem jede Relation R aus z_ω durch eine bezüglich der zugehörigen Konzepte mögliche Relation zu R ersetzt wird.

Da in der Menge der Erweiterungen einer ω-Relation ihre Vervollständigungen enthalten sind, erhalten wir für jeden ω-Zustand z_ω eine Obermenge von $POSS_c(z_\omega)$ als Menge möglicher Zustände.

Betrachte die erste Relation **GAST** aus Abschnitt 7.1:

GAST

NAME	ORT	BEGLEITUNG
Harald	Toulouse	Elvira
Gisela	Bonn	-

Diese Relation ist eine mögliche Relation bezüglich der beiden Konzepte "x ist Gast aus o" und "x kommt in Begleitung von y". Das partielle Tupel der Relation enthält eine sinnvolle Aussage bezüglich des ersten Konzeptes und keine Aussage, die sich auf das zweite Konzept bezieht.

Für den Zustand aus Abschnitt 7.1 mit den Relationen **GAST** und **BEGLEITUNG** ist (Gisela, -) kein erlaubtes Tupel, da es kein Konzept gibt, das nur aus dem Attribut NAME besteht. Daher ist die angegebene Relation **BEGLEITUNG** keine mögliche Relation zu sich selbst.

In der Relation **BEGL_BUFFET** aus Abschnitt 7.1 ist das Tupel (Marga, -, -) kein erlaubtes Tupel, da das Tupel für beide angegebenen Konzepte in einem obligatorischen Attribut nicht definiert ist.

Es bietet sich an, als Semantik partieller DB-Zustände wie bisher die Menge aller zugehörigen möglichen DB-Zustände zu betrachten. Damit lassen sich über diese Menge auch wieder Formen gesicherter Antworten definieren. In ihnen ist nur Information enthalten, die in allen möglichen Zuständen gültig ist. Mit unserer Festlegung von möglichen Zuständen, die die CWA zu "retten" versucht, können wir dabei allerdings in einen gewissen Widerspruch zur natürlichen Auffassung von Anfrageformulierungen in formalen Anfragesprachen geraten. Betrachten wir die TRC-Anfrage (x)/ BEGL_BUFFET x. Offensichtlich ist der Inhalt der Relation BEGL_BUFFET aus dem gegebenen Zustand eine gesicherte Antwort auf diese Anfrage. Da das Tupel (Marga, -, -) aber in einer der möglichen Relationen bzgl. der angenommenen Konzeptmenge nicht enthalten ist, muß es in einer gesicherten Antwort fehlen. Konzepte müssen daher auch beim Formulieren von Anfragen eine Rolle spielen.

7.3 Gesicherte Antworten auf TRC-Anfragen

Für unbekannte Attributwerte kann zur Konkretisierung der allgemein gehaltenen Festlegung gesicherter Information in Antworten auf die Definition von Antworten für gewöhnliche, d.h. totale Zustände zurückgegriffen werden. Wie wir gesehen haben, ist dies nicht ohne weiteres mehr möglich für die in diesem Kapitel betrachtete Interpretation fehlender Attributwerte. Es sind im allgemeinen mögliche Zustände zu berücksichtigen, in denen fehlende Attributwerte mit der Interpretation "kein Wert vorhanden" vorkommen.

Das Wissen "kein Wert vorhanden" ist vollständiges (deterministisches) Wissen in dem Sinn, daß es keine Alternativen zuläßt. Die übliche Vorgehensweise für die Auswertung von Anfragen ist trotzdem nicht ohne weiteres anwendbar, da kein definierter Wert des zugehörigen Wertebereichs vorliegt, sondern Wissen *über* einen Wert (seine Nicht-Existenz). Dadurch können TRC-Anfragen und Anfragen in der Relationenalgebra "unterspezifiziert" sein. Außerdem sind die Probleme im

Zusammenhang mit der CWA, die wir im vorigen Abschnitt mit Hilfe der Konzepte teilweise lösen konnten, nicht mehr so einfach zu behandeln. Betrachten wir dazu einige Beispiele.

Sei folgende Anfrage an das Schema {(**GAST**(....), **BEGLEITUNG**(....)} von oben gerichtet:

"*Gib die Daten aller Gäste, die nicht mit Elvira als Begleitung kommen.*"

Im TRC bietet sich als Formulierung an:

(x)/ GAST x $\wedge$ $\neg$ ($\exists$y: BEGLEITUNG y)(y.NAME = x.NAME $\wedge$ y.B_NAME = Elvira)

In jedem möglichen Zustand zu dem in Abschnitt 7.1 angegebenen Zustand (S. 157) ist die Relation **BEGLEITUNG** total (sie enthält nur das Tupel (Harald, Elvira)), daher entstehen keine Probleme bei der Auswertung. Implizit ist mit dieser Formulierung die Anfrage wie folgt ergänzt worden:

"*Gib die Daten aller Gäste, die in Begleitung einer Person kommen, deren Name nicht Elvira ist, oder die ohne Begleitung kommen.*"

Betrachte nun die entsprechende an das Schema {(**GAST**(....), **BEGL_BUFFET**(....)} gerichtete Formulierung:

(x)/ GAST x $\wedge$ $\neg$ ($\exists$y: BEGL_BUFFET y)(y.NAME = x.NAME $\wedge$ y.B_NAME = Elvira)

Mit der Interpretation "Wert nicht vorhanden" für alle Vorkommen fehlender Werte in dem entsprechenden Zustand in Abschnitt 7.1 (S. 159) erhalten wir als möglichen Zustand die angegebene **GAST**-Relation zusammen mit den ersten beiden Tupeln der Relation **BEGL_BUFFET**. Wenn wir bei der Auswertung die Belegung von Variablen mit partiellen Tupel zulassen und dem Vergleich ω = Elvira den Wert *falsch* zuordnen, erhalten wir ein Ergebnis, das zu der ergänzten Anfrageform paßt. Der Effekt ist der gleiche, als wenn die Belegung von y mit dem Tupel (Gisela, ω, Salat) nicht vorgenommen wird. Ordnen wir dagegen dem Vergleich den Wert *undefiniert* zu und verwenden wir zur Auswertung die dreiwertige Logik, dann qualifizieren sich zwar die Tupel (Frank, Kiel) und (Marga, Trier) für die Antwort, nicht dagegen das Tupel (Gisela, Bonn). Die erhaltene Antwort ist in diesem Fall für die angegebene Bedeutung nicht vollständig.

In der Anfrage wird nur Bezug auf das Konzept "Gast x kommt in Begleitung von y" genommen. Daher erscheint es naheliegend, für eine Auswertung zu verlangen, daß Tupel, die für dieses Konzept keine Aussage enthalten, als nicht vorhanden angesehen werden. Dieser Effekt wird für die Beispielanfrage durch die Wertzuordnung ω = Elvira $\equiv$ *falsch* erreicht. Eine solche Wertzuordnung führt aber nicht immer zum gewünschten Effekt. Dies zeigt folgende TRC-Anfrage an das gleiche Schema:

(x)/ GAST x $\wedge$ ($\exists$y: BEGL_BUFFET y)(y.NAME = x.NAME $\wedge$ $\neg$ (y.NAME = Elvira))

Auch in dieser Formulierung wird nur das erwähnte Konzept angesprochen. Ordnen wir dem Vergleich ω = Elvira den Wert *falsch* zu, dann qualifiziert sich das Tupel (Gisela, Bonn); dagegen qualifizieren sich nicht die Tupel (Frank, Kiel) und (Marga, Trier). Es wird also unterschieden zwischen Tupeln, zu denen aufgrund der CWA eine negative Aussage in bezug auf das Konzept gehört, und Tupeln, die sich nicht qualifizieren, weil die mit ihnen verknüpften Tupel keine Aussage beitragen.

Offensichtlich ist eine Wertzuordnung in Abhängigkeit von der Struktur einer Anfrage vorzunehmen, um den Effekt, die Gleichstellung von Information, die

aufgrund der CWA gefolgert werden kann, mit Information, die explizit mit Hilfe undefinierter Werte repäsentiert wird, erzielen zu können. Wir werden dies hier nicht weiter verfolgen. Es erscheint dazu notwendig, Zusammenhänge zwischen natürlichsprachlichen und formalen Anfrageformulierungen genauer zu untersuchen (für eine Diskussion solcher Zusammenhänge siehe etwa [BS76]).

7.3.1 Das Verfahren von N. Lerat und W. Lipski

In [LeLi86] wird ein Verfahren vorgestellt, das spezielle Anfragen in einer verallgemeinerten Variante des DRC an Universalrelationen auf eine Menge von Anfragen an ein zugehöriges DB-Schema σ zurückführt. Dabei wird von Universalrelationen vorausgesetzt, daß zu jedem ihrer Tupel ein Relationstyp in σ existiert, dessen Attributmenge mit der Menge von Attributen übereinstimmt, in der das Tupel definiert ist. Der Zustand zu σ, der einer gegebenen Universalrelation entspricht, besteht aus den totalen Projektionen der Universalrelation auf die Relationstypen von σ. Unter der Annahme, daß die Relationstypen einzelne Konzepte modellieren und Konzepte nur obligatorische Attribute haben, stellen die betrachteten Universalrelationen damit in unserer Betrachtungsweise bezüglich der Konzepte erlaubte Relationen dar.

Die Verallgemeinerung des wertebereichsorientierten Kalküls besteht darin, daß zwei Typen von Variablen zugelassen werden: Variable, die nur mit definierten Werten belegt werden dürfen und Variable, deren Belegung mit dem speziellen Wert *undefiniert* (-) erlaubt ist. Die Idee ist hierbei auch, partielle Tupel in Antworten zuzulassen, wenn der Qualifikationsausdruck der Anfrage diese Tupel nicht explizit ausschließt. In Vergleichsausdrücken, in denen ein undefinierter Attributwert vorkommen darf, werden nur die Vergleichsoperatoren = und $\neq$ zugelassen. Der Wert *undefiniert* wird jedem Wertebereich hinzugefügt und wie eine gewöhnliche Konstante betrachtet, die von allen anderen Konstanten eines Wertebereichs verschieden ist. Bei der Auswertung einer Anfrage werden die Belegungsmöglichkeiten betrachtet, die sich durch sukzessive Ersetzung aller Variablen des neuen Typs durch eine gewöhnliche Variable oder den Wert *undefiniert* ergeben. Gewöhnliche Variable werden nur mit gewöhnlichen Werten belegt und die Prädikate werden den Relationstypen von σ angepaßt, indem alle Vorkommen von *undefiniert* gestrichen werden. Mit dieser Vorgehensweise wird die Auswertung einer Anfrage an eine Universalrelation durch Überführung in die Auswertung einer Menge von Anfragen an eine Zerlegung der Universalrelation vorgenommen.

Das in [LeLi86] vorgestellte Verfahren ist nur für einen Kalkül angegeben, in dem keine Vergleichsausdrücke vorkommen. Im folgenden Beispiel haben wir eine sinngemäße Erweiterung auf Anfragen mit Vergleichsausdrücken vorgenommen.

Beispiel 7.1: Betrachte folgende TRC-Anfrage an unser letztes Beispielschema. Die Variable, die dem Attribut B_NAME zugeordnet ist, darf mit *undefiniert* belegt werden, was durch Großschreibung kenntlich gemacht ist.

(x,Y,z)/ BEGL_BUFFET(x,Y,z) $\wedge$ Y $\neq$ Elvira

Es sind im ersten Schritt zwei Anfragen zu erzeugen:

(x,y,z)/ BEGL_BUFFET(x,y,z) $\wedge$ y $\neq$ Elvira und

(x,-,z)/ BEGL_BUFFET(x,-,z) $\wedge$ - $\neq$ Elvira

Im nächsten Schritt wird die zweite Formulierung umgeformt in

(x,z)/ BUFFET(x,z) $\wedge$ - $\neq$ Elvira

Beim Vereinigen der Ergebnisse aller Teilanfragen werden die Typen der Ergebnistupel durch Hinzufügen von *undefiniert* angepaßt.
Für unsere Beispielrelation steuert

(x,y,z)/ BEGL_BUFFET(x,y,z) $\wedge$ y $\neq$ Elvira

nichts zum Ergebnis bei, da die Relation kein totales Tupel enthält.

(x,z)/ BUFFET(x,z) $\wedge$ - $\neq$ Elvira

liefert (Gisela, -, Salat).
Ändern wir die Variable z in der Ausgangsformulierung in Z ab, dann erhalten wir auch das Tupel (Marga, -, -) in der Antwort. Voraussetzung ist hierbei aber, daß {NAME} ein Konzept darstellt. Diese Forderung ist für unser Beispielschema nicht sinnvoll.

Teilausdrücke, in denen Variable neuen Typs quantifiziert werden, führen ebenfalls zum Aufspalten von Anfragen. So wird ein Ausdruck der Form

$(\exists X)(F(X))$

überführt in

$(\exists x)(F(x)) \vee F(-)$.

Das folgende Beispiel zeigt, daß dieses Vorgehen nicht einfach auf Anfragen mit Vergleichsausdrücken übertragen werden kann.

Beispiel 7.2: Betrachte wieder das **GAST/BEGLEITUNG**-Schema aus Abschnitt 7.1 mit dem Zustand

GAST

NAME	ORT
Harald	Toulouse
Gisela	Bonn
Frank	Kiel

BEGLEITUNG

NAME	B_NAME
Harald	Elvira
Gisela	-

Sei folgende DRC-Anfrage gegeben:

(x,y)/ GAST(x,y) $\wedge$ $(\exists Z)$(BEGLEITUNG(x,Z) $\wedge$ Z $\neq$ Elvira)

Mit der Vorschrift von oben erhalten wir die beiden folgenden Anfragen:

(*) (x,y)/ GAST(x,y) $\wedge$ $(\exists z)$(BEGLEITUNG(x,z) $\wedge$ z $\neq$ Elvira)

(x,y)/ GAST(x,y) $\wedge$ BEGLEITUNG(x,-) $\wedge$ - $\neq$ Elvira

Die letzte Anfrage wird umgeformt in

(**) (x,y)/ GAST(x,y) $\wedge$ BEGL_NAME(x) $\wedge$ - $\neq$ Elvira.

Dabei ist mit BEGL_NAME eine Relation bezeichnet, von der angenommen wird, daß sie durch Projektion von BEGLEITUNG auf das Attribut NAME erhalten werden kann. Die Antwort auf die mit (*) gekennzeichnete Anfrage ist leer. Für die mit (**) markierte Anfrage und damit als Gesamtergebnis erhalten wir dagegen {(Harald, Toulouse), (Gisela, Bonn)}, da die Bedingung - $\neq$ Elvira zu *wahr* ausgewertet und die Enthaltenseinsbedingung bei der Auswertung vorausgesetzt wird. Das Tupel (Harald, Toulouse) paßt aber offensichtlich nicht zur Anfrage.

Die beiden Beispiele zeigen, daß die Methode von Lerat und Lipski nicht ohne weiteres auf beliebige DB-Schemata und beliebige Anfragen übertragen werden kann. Ein Nachteil der Methode für ihre praktische Verwendung ist außerdem die mögliche Anzahl der Teilfragen, die bei der Zerlegung erzeugt werden. Sie ist exponentiell in der Anzahl der Variablen, die mit *undefiniert* belegt werden dürfen.

7.3.2 Effizienzprobleme

In [LeLi86] wird vorgeschlagen, zur Auswertung von Anfragen an DB-Zustände mit Vorkommen von "no information" Attributwerten zunächst die "Methode von Lipski" (vermutlich sind hier die Vorschläge in [Lip79] und [Lip81] gemeint) anzuwenden, um gesicherte Ergebnisse für die Interpretation "Attributwert existiert, ist aber nicht bekannt" zu erhalten und anschließend das vorgestellte Verfahren für die Interpretation "non applicable" anzuwenden. Da die Vorschläge von Lipski sich nur auf die Selektionsoperation beziehen, erhalten wir damit schon aus diesem Grund kein allgemeines Verfahren. Außerdem sind die Bemerkungen von oben zu beachten. Es gibt noch einen anderen Grund, warum wir diese Vorgehensweise im allgemeinen nicht anwenden können: Die "no information"-Interpretation bedeutet, daß Vorkommen fehlender Attributwerte beliebig mit den beiden unterlegbaren Bedeutungen versehen werden können, was wir bei der Definition möglicher Relationen und Zustände beachtet haben. Damit führt eine Auswertung getrennt nach den beiden Bedeutungen im allgemeinen nicht zum gleichen Ergebnis wie eine Auswertung mit Berücksichtigung aller Möglichkeiten. Dies zeigt ein einfaches Beispiel.

Beispiel 7.3: Sei folgende TRC-Anfrage an das Schema {(**GAST** (NAME, ORT), **BEGLEITUNG** (NAME, B_NAME)} gegeben:

$$(x)/\ \text{GAST}\ x \wedge (\neg\ (\exists y\colon \text{BEGLEITUNG}\ y) \vee (\exists w\colon \text{BEGLEITUNG}\ w)(w.\text{NAME} = x.\text{NAME} \wedge (\exists z\colon \text{BEGLEITUNG}\ z)(z.\text{NAME} \neq w.\text{NAME})))$$

Wir betrachten drei Formen möglicher Zustände zu dem Basiszustand, der aus dem in Abschnitt 7.1 (und Beispiel 7.2) angebenen Zustand zu dem Schema erhalten wird, indem Elvira in der Relation **BEGLEITUNG** durch "-" ersetzt wird. Da die **GAST**-Relation in diesem Basiszustand total ist, genügt es, sie einmal anzugeben. Die möglichen Zustände werden allein durch die Relationen vom Typ **BEGLEITUNG** bestimmt.

GAST

NAME	ORT
Harald	Toulouse
Gisela	Bonn
Frank	Kiel

BEGLEITUNG

NAME	B_NAME
Harald	Elvira
Gisela	Hans

1

BEGLEITUNG

NAME	B_NAME
Harald	Elvira

2

BEGL.

Φ

3

Der erste Zustand ist ein möglicher Zustand, der für die Bedeutung "Wert unbekannt" der beiden fehlenden Attributwerte erhalten werden kann, der zweite Zustand wird erhalten mit der Bedeutung "Wert nicht vorhanden" für den fehlenden Wert im ersten Tupel und "Wert unbekannt" im zweiten Tupel. Im dritten möglichen Zustand ist die Relation **BEGLEITUNG** leer, da hier beide Vorkommen von "-" als "Wert nicht vorhanden" interpretiert werden.
Da wir nur totale Relationen in den drei möglichen Zuständen haben, ist das Ergebnis der TRC-Anfrage wie im Standardfall bestimmt. Wir erhalten:

1: {(Harald, Toulouse), (Gisela, Bonn)}

2: Φ

3: {(Harald, Toulouse), (Gisela, Bonn), (Frank, Kiel)}

Die gesicherte Antwort ist damit leer. Dieses Ergebnis kann offensichtlich nicht erhalten werden, indem nur die möglichen Zustände betrachtet werden, die durch Interpretation der fehlenden Werte entweder als "Wert unbekannt" oder als "Wert nicht vorhanden" erhalten werden (Zustände 1 und 3).

Das Beispiel zeigt, daß es Schemata und Zustände gibt, zu denen Anfragen konstruiert werden können mit folgender Eigenschaft: Bei der Bestimmung gesicherter Information sind beliebige mögliche Zustände zu berücksichtigen. Das gesicherte Ergebnis kann in Abhängigkeit davon, wieviele und welche der fehlenden Attributwerte als "Wert nicht bekannt" bzw. als "Wert nicht vorhanden" interpretiert werden, leer sein oder nicht. Damit kann kein allgemeines, nicht triviales Verfahren effizient sein. Unter einem allgemeinen Verfahren verstehen wir dabei ein Verfahren, das nicht für spezielle Mengen von Anfragen und/oder Zustände konstruiert ist; ein Verfahren wird als trivial bezeichnet, wenn es immer die leere Antwort liefert.

Um zu einem effizienten, allgemeinen und nicht trivialen Verfahren zu gelangen, müssen wir die Definition gesicherter Antworten ändern. Dies kann über die Änderung der Definition möglicher Zustände oder der Definition gesicherter Antworten als (in der Regel unvollständige) Beschreibung der Menge gesicherter Antworttupel erfolgen. Eine weitere Möglichkeit besteht darin, nur Basiszustände zuzulassen, in denen alle Tupel erlaubt sind, wenn ihre fehlenden Attributwerte als "Wert nicht vorhanden" interpretiert werden (wie oben schon erwähnt, wird diese Annahme in [LeLi86] gemacht). Damit sind aber Schemata wie unser GAST/BEGLEITUNG-Beispiel nicht mehr sinnvoll für die "no information"-Interpretation fehlender Attributwerte geeignet.

Die Definition gesicherter Antworten über die Betrachtung aller möglichen Zustände erscheint uns natürlich und sollte nicht geändert werden. Es bleibt demnach die Änderung der Definition möglicher Zustände. Da die Auswirkungen der CWA ohnehin nur teilweise bei der gegebenen Definition berücksichtigt wurden, erscheint dieser Weg am naheliegendsten. Er führt zu einer Interpretation von Anfragen, die mehr an der Darstellung der Information in der Datenbank orientiert ist. Eine solche mehr "syntaktische" Sichtweise wird gelegentlich auch für die Betrachtung von SQL-Anfragen insbesondere im Zusammenhang mit Nullwerten empfohlen ("*Our advice is to take care that you use the SQL definition of a word rather than the standard English definition*", [MS93], S. 35). Da wir im folgenden Kapitel einen konkreten Vorschlag für eine Auswertung von Anfragen mit dieser Sichtweise machen, wollen wir hier nicht weiter auf Auswertungsverfahren für TRC-Anfragen eingehen.

7.4 Der Vorschlag von C. Zaniolo

In [Zan82] und ausführlicher in [Zan84] wird ein Vorschlag zur Auswertung von Algebra- und TRC-Anfragen in partiellen DB-Zuständen gemacht, wobei fehlenden Attributwerten die hier betrachtete "no information"-Interpretation zugrunde gelegt wird. Dieser Vorschlag ist sicher der am häufigsten genannte, wenn es um die "no information"-Interpretation geht. Ein Teil der Ideen scheint auch Eingang in das Nullwertkonzept von SQL gefunden zu haben. Weshalb wir uns mit diesem Vorschlag trotzdem nur am Rande beschäftigen, wollen wir im folgenden erläutern.

Die Operationen der Relationenalgebra sind in [Zan84] auf Äquivalenzklassen von partiellen Relationen definiert. Zur Modellierung partieller Relationen, wie sie dort betrachtet werden, können ω-Relationen hergenommen werden. Die Äquivalenzklassen lassen sich mit Hilfe einer verallgemeinerten Form der Überdeckungsäquivalenz von ω-Relationen definieren, wie wir sie in Kapitel 2 definiert haben. Die Verallgemeinerung besteht darin, daß bei Überdeckungen nicht mehr die Typ-Gleichheit von Relationen bzw. Tupeln verlangt wird. Diese Verallgemeinerung ist aber für unsere Betrachtungen ohne Bedeutung. Zwei Relationen R und S gehören damit zur gleichen Klasse, wenn sie sich gegenseitig überdecken. Wie wir in Kapitel 2 gesehen haben, bedeutet dies, daß mit einer Relation R auch alle Relationen in der durch R bestimmten Klasse <R> enthalten sind, die die gleichen maximalen Elemente wie R haben. Damit erhalten wir als minimalen Repräsentanten der Klasse <R> die Relation, die genau die maximalen Elemente enthält, und als maximalen Repräsentanten die Relation mit allen Tupeln, die in dem durch die maximalen Elemente bestimmten Halbverband enthalten sind.

In [Kel86] wird gezeigt, daß keine Relationenalgebra definiert werden kann, die auf diesen Äquivalenzklassen operiert und mit der gewöhnlichen Mengentheorie verträglich ist. So gilt im allgemeinen etwa nicht $R \setminus (R \setminus S) = R \cap S$. Die Nicht-Gültigkeit derartiger Gleichungen haben wir auch bei den Operationsdefinitionen in Kapitel 4 und 5 in Kauf genommen, um Effizienz zu erreichen. So gilt dort zum Beispiel im allgemeinen nicht $R \cap R = R$, wenn die Struktur eines Ausdrucks bei seiner Auswertung nicht mit einbezogen wird. Das Problem mit dem Ansatz von Zaniolo ist aber schwerwiegender, da im allgemeinen nicht mehr garantiert ist, daß die Antworten gesicherte Information in der von uns verwendeten Bedeutung darstellen, und da es auch keine Möglichkeit gibt, Operationen so auf den eingeführten Äquivalenzklassen zu definieren, daß diese Eigenschaft garantiert ist. Darüberhinaus werden Annahmen über die Gleichheit der Bedeutung fehlender Attributwerte gemacht, die mit der Auffassung von "Wert unbekannt" oder "Wert nicht vorhanden" nicht im Einklang stehen. Als Ergebnis des Schnitts zweier Relationen R und S verträglichen Typs mit dem gleichen Inhalt $\{(a, \omega)\}$ erhalten wir zum Beispiel diesen Inhalt als Ergebnis.

Um zu zeigen, daß mit dem Ansatz von Zaniolo im allgemeinen auch bei Änderung der Operationsdefinitionen, etwa durch die Unterscheidung in s- und m-Varianten, keine gesicherten Ergebnisse erhalten werden können, betrachten wir ein einfaches Beispiel.

Beispiel 7.4: Seien $R = \{(a,b)\}$ und $S = \{(a,\omega)\}$ zwei ω-Relationen über der gleichen Attributmenge {A,B}. Alle Relationen in der durch S bestimmten Äquivalenzklasse sind in der Klasse zu R enthalten. Als Ergebnis der Vereinigung $R \cup S$ erhalten wir daher R. Betrachten wir den Ausdruck $S \setminus (R \cup S)[B \neq b]$. Als Ergebnis ergibt

sich mit den Definitionen von [Zan84] die Relation S. Da gemäß [Zan84] alle Operationen auf den minimalen Elementen der Äquivalenzklassen definiert werden können, d.h. nur die maximalen Elemente des entsprechenden Halbverbandes berücksichtigt werden müssen, kann durch keine Operationsdefinition die in dem Tupel aus S enthaltene Information berücksichtigt werden. Dazu müßte von der Klassenbetrachtung abgewichen werden.

8 Nullwerte in SQL

Nachdem wir bisher die Modellierung unvollständiger Information sowie Auswertungs- und Äquivalenzfragen auf der Ebene des ursprünglichen relationalen Datenmodells betrachtet haben, wollen wir uns in diesem Kapitel der in der Praxis zum de facto-Standard gewordenen Sprache SQL zuwenden. Für diese Sprache wurde im zweiten Standard ([ISO89]) eine Möglichkeit geschaffen, Tupel in entsprechend gekennzeichneten Attributen ohne konkreten Wert zu belassen. Diese Möglichkeit ist eine dringende Forderung aus der Praxis und wurde in einigen frühen Datenbanksystemen, etwa in CODASYL-Systemen, durch die Erlaubnis, Felder mit dem speziellen COBOL-Wert *low-value* zu versorgen, realisiert. Die Zuordnung einer Bedeutung für ein mit einem solchen Wert belegtes Feld ergibt sich bei diesen Systemen allein aus der Form der Berücksichtigung des Wertes in Anwendungsprogrammen, d.h. ein solcher Wert wird von den Systemen selbst wie alle übrigen Werte auch als gewöhnlicher programmiersprachlicher Wert betrachtet. So ergibt sich für einen Vergleich einer Konstanten mit einem Vorkommen von *low-value* stets ein definiertes Ergebnis und zwar dasjenige Ergebnis, das in der Programmiersprache COBOL für diesen Fall festgelegt ist. Ein solcher Wert erfüllt damit die Funktion eines systemdefinierten Default-Wertes. Einer Modellierung unvollständiger Information mit Hilfe von Default-Werten, die allerdings vom Benutzer festgelegt werden sollen, wird zum Beispiel in [Dat86] der Vorzug gegeben.

Bei der Einführung "fehlender Werte" in SQL wurde ein anderer Weg beschritten, der sich an der im vorangehenden Kapitel beschriebenen Vorgehensweise orientiert und damit auch die hierfür aufgezeigten Probleme mit sich bringt. Ein *Nullwert* in SQL (dargestellt durch das reservierte Wort NULL) ist nach Definition ([ISO89], [ISO92])

a special value, or mark, that is used to indicate the absence of any data value.

Es werden keine weiteren Annahmen bezüglich der Bedeutung eines Nullwertes gemacht, d.h. es ist bei der Definition eines DB-Schemas auch nicht möglich, für einzelne Attribute eine gewünschte Interpretation für Vorkommen von Nullwerten unter dem Attribut anzugeben (Änderungen in dieser Hinsicht sind für kommende SQL-Standards zu erwarten ([MS93])).

Zur Berücksichtigung von Nullwerten werden bei der Festlegung der Semantik von SQL im wesentlichen die folgenden Erweiterungen vorgenommen (auf weitere Besonderheiten gehen wir im nächsten Abschnitt ein):

- Alle Vergleiche, in denen NULL als Operand vorkommt, erhalten den Wert *unbekannt.*
- Bei der Auswertung von WHERE-Klauseln eine dreiwertige Logik verwendet.
- In das Ergebnis von Anfragen und Teilanfragen werden nur solche Tupel aufgenommen, für die die Auswertung der zugehörigen WHERE-Klausel den Wert *wahr* liefert, d.h. es wird die gesicherte Antwort einer solchen Anfrage bzw. Teilanfrage berechnet.

Wie wir im vorigen Kapitel gesehen haben und wie aus unseren Betrachtungen im vierten und fünften Kapitel folgt, führt eine solche Vorgehensweise bei der

Relationenalgebra und beim TRC im allgemeinen zu Antworten, die keine gesicherte Information darstellen. Da SQL prädikative Konstrukte und insbesondere die Möglichkeit zur Negation von Prädikaten bereitstellt, folgt damit aus diesen Betrachtungen, daß SQL-Antworten im allgemeinen keine gesicherten Antworten sind. Darüberhinaus können Äquivalenzen, die gemäß der Prädikatenlogik und der verwendeten dreiwertigen Logik gelten müßten, für bestimmte Anfragen und Zustände nicht gültig sein. Auf solche Äquivalenzprobleme wurde schon in [Dat90] hingewiesen; eine mehr formale Behandlung des Problems findet sich in [NPS91].

Wir wollen im folgenden zunächst an Beispielen demonstrieren, welche Probleme die SQL-Semantik im Hinblick auf die Behandlung fehlender Attributwerte in sich birgt. Anschließend machen wir einen Vorschlag, wie gesicherte Antworten für SQL-Anfragen so definiert werden können, daß eine effiziente Berechnung möglich ist. Wir geben dann ein Verfahren zur Ergänzung und Modifikation von SQL-Anfragen an, durch das ohne Änderung der SQL-Semantik gewährleistet werden kann, daß Antworten stets gesicherte Antworten gemäß der gegebenen Definition darstellen. Dieses Verfahren wurde in gekürzter Form schon in [Kln94] vorgestellt.

8.1 Probleme mit der Semantik

Sei die folgende Relation (in SQL: Tabelle) gegeben:

PERSON

NACHNAME	VORNAME	GEBURTSTAG
Schmidt	Richard	NULL
Schmidt	Willi	19.1.

{NACHNAME, VORNAME} soll Schlüssel des Relationstyps PERSON sein. In Anlehnung an das bekannte "Geburtstagsparadoxon" formulieren wir die folgende Anfrage Q1:

> "*Gib die Daten aller Personen, zu denen es keine andere Person mit gleichem Nachnamen, aber anderem Vornamen gibt, die am gleichen Tag Geburtstag hat*"

Eine naheliegende Formulierung dieser Anfrage in SQL lautet wie folgt:

```
SELECT *
FROM  PERSON x
WHERE NOT EXISTS
                (SELECT *
                 FROM  PERSON y
                 WHERE x.NACHNAME  = y.NACHNAME
                    AND  x.VORNAME  <> y.VORNAME
                    AND  x.GEBURTSTAG = y.GEBURTSTAG)
```

Gemäß der SQL-Semantik besteht die Antwort auf diese Anfrage aus allen Tupeln der Relation PERSON: Für alle Belegungsmöglichkeiten, die sich für x und y ergeben, hat entweder einer der Teilterme den Wert *falsch* (x.VORNAME <> y.VORNAME) oder den Wert *unbekannt* (x.GEBURTSTAG = y.GEBURTSTAG); damit

hat die WHERE-Klausel stets den Wert *falsch* oder *unbekannt* (Anwendung der dreiwertigen Logik) und die Teilanfrage liefert für jede Belegung der freien Variablen x die leere Menge als Ergebnis (es qualifizieren sich nur Tupel für eine Teilanfrage, für die die WHERE-Klausel *wahr* als Wert hat). Der "Quantor" EXISTS angewandt auf eine Teilanfrage liefert *falsch* als Ergebnis, wenn die Antwort auf die Teilanfrage leer ist. Dies führt letztlich zur angegebenen Antwort.

Mit der Bedeutung "Wert existiert, ist aber nicht bekannt" von Nullwerten im Attribut GEBURTSTAG, die bei natürlichen Personen sicherlich angebracht ist, muß die SQL-Antwort als nicht gesicherte Information angesehen werden. Sie repräsentiert eine Information zur Anfrage, die nicht mit Sicherheit aus den vorhandenen Informationen geschlossen werden kann. Falls der Geburtstag von Richard Schmidt der gleiche ist wie der von Willi Schmidt, liefert die Anfrage an einen entsprechenden totalen Zustand die leere Antwort, d.h. die von SQL gelieferte Antwort paßt für diesen Fall nicht zur Anfrageformulierung. Sie muß daher als *unsichere* oder *mögliche* Antwort angesehen werden.

Die SQL-Antwort paßt jedoch zu einer etwas abgeänderten Formulierung der Anfrage Q1, in der sich die Auswertungsmethode von SQL direkt widerspiegelt:

> "*Gib die Daten aller Personen, so daß es keine andere Person gibt, die mit Sicherheit den gleichen Nachnamen und den gleichen Geburtstag hat.*"

Betrachten wir eine andere Möglichkeit zur Formulierung von Q1:

```
SELECT *
FROM PERSON x
WHERE x.NACHNAME <> ALL
            (SELECT y.NACHNAME
             FROM PERSON y
             WHERE x.GEBURTSTAG = y.GEBURTSTAG
                AND x.VORNAME <> y.VORNAME)
```

Für diese Formulierung erhalten wir die gleiche Antwort wie für die erste Formulierung: Es qualifizieren sich wiederum für beide Belegungsmöglichkeiten von x keine Tupel für die Teilanfrage; wird für die universell quantifizierte Teilanfrage in einem Vergleichsausdruck aber eine leere Relation erhalten, dann hat der Ausdruck den Wert *wahr*.

In der Booleschen Logik wie in der dreiwertigen Logik (nach Kleene, nicht z.B. nach Lukasiewicz) gilt die folgende Äquivalenz für beliebige Terme A, B und C:

$$A \wedge \neg B \Rightarrow \neg C \quad \equiv \quad C \wedge \neg B \Rightarrow \neg A$$

Deshalb sollte die folgende Anfrageformulierung das gleiche Ergebnis liefern wie die vorangehende Formulierung:

```
SELECT *
FROM PERSON x
WHERE x.GEBURTSTAG <> ALL
            (SELECT y.GEBURTSTAG
             FROM PERSON y
             WHERE x.NACHNAME = y.NACHNAME
                AND x.VORNAME <> y.VORNAME)
```

Die SQL-Antwort zu dieser Formulierung von Q1 ist aber eine leere Relation. Der Grund hierfür liegt in der Behandlung von Nullwerten in quantifizierten Teilausdrücken. Es qualifiziert sich für jede Belegung von x ein Tupel für die Teilanfrage; dieses Tupel oder die Belegung von x hat aber im Attribut Geburtstag NULL als Wert. Damit kann für jede Belegung von x dem Vergleich mit dem Ergebnis der Teilanfrage weder der Wert *wahr* noch der Wert *falsch* zugeordnet werden. In diesem Fall legt die SQL-Semantik *unbekannt* als Wert für den quantifizierten Teilausdruck fest. Somit wird Q1 durch diese Formulierung wie folgt interpretiert:

> "*Gib die Daten aller Personen, so daß es mit Sicherheit keine andere Person gibt, die den gleichen Nachnamen und den gleichen Geburtstag hat*".

Alle angegebenen SQL-Formulierungen der Anfrage Q1 sind naheliegende Formulierungen dieser Anfrage. Es ist daher unbefriedigend, daß unterschiedliche Antworten erhalten werden können. Solche unterschiedlichen Antworten sind auch nicht mit einer anderen Interpretation der Vorkommen von NULL akzeptabel. Betrachten wir im folgenden die Interpretation fehlender Attributwerte, die wir schon im vorangehenden Kapitel untersucht haben:

Ein fehlender Wert in einem Attribut eines Tupels existiert auch nicht für den Gegenstand, der durch das Tupel modelliert wird.

Als Beispiel betrachten wir eine Relation, in der Information über Aufträge enthalten ist. Zu jedem Auftrag ist eine Auftragsnummer (AUFNR), ein Datum für den Auftragseingang und die Höhe des Auftrags, sein Volumen, vermerkt. {AUFNR} soll Schlüssel sein. Zusätzlich soll in Tupeln der Relation die Nummer des Angestellten vermerkt sein, der den Auftrag bearbeiten soll. Es ist möglich, daß in bestimmten Bearbeitungsphasen, die ein Auftrag nach Eingang durchläuft, niemand für seine Bearbeitung zugeteilt ist. In diesen Fällen wird der Wert NULL für das Attribut BEARB eingetragen, um die fehlende Beziehung zwischen einem Auftrag und einem Bearbeiter zu repräsentieren.

AUFTRAG

AUFNR	DATUM	VOLUMEN	BEARB
1	3.11.	10000	NULL
3	3.11.	5500	A2
4	4.11.	3000	NULL

Die Information, die durch diese Relation dargestellt wird, ist vollständig: Die beiden Vorkommen von NULL repräsentieren die Information, daß kein Bearbeiter für den entsprechenden Auftrag eingeteilt ist.

Die folgende Anfrage Q2 ist analog zur Anfrage Q1 aufgebaut.

Q2:

> "*Gib die Daten jedes Auftrags, so daß es keinen Angestellten gibt, der diesen Auftrag und einen weiteren Auftrag mit dem gleichen Eingangsdatum bearbeitet*".

Die nachfolgende SQL-Formulierung von Q2 entspricht der ersten Formulierung, die wir oben für Q1 angegeben haben:

```
SELECT *
FROM AUFTRAG x
WHERE NOT EXISTS
            (SELECT *
             FROM AUFTRAG y
             WHERE  x.DATUM = y.DATUM
                AND  x.AUFNR <> y.AUFNR
                AND  x.BEARB = y.BEARB)
```

Für keine Belegung von x qualifiziert sich eine Belegung von y für die Teilanfrage, da die Auswertung des Qualifikationsteils der Teilanfrage entweder *falsch* oder *unbekannt* liefert. Damit erhalten wir wie oben alle Tupel der Relation als Antworttupel. Mit der hier für die Vorkommen von NULL gewählten Interpretation stellt diese Antwort aber im Gegensatz zu oben eine gesicherte Information dar: Da nur der Angestellte A2 einen Auftrag bearbeitet, ist die in der Anfrage Q2 formulierte Bedingung für alle Tupel erfüllt. Ändern wir die Interpretation der Vorkommen von NULL in *Wert vorhanden, aber unbekannt* ab, dann stellt die Antwort keine gesicherte Information dar, da A2 neben Auftrag 3 auch Auftrag 1 bearbeiten könnte und beide Aufträge das gleiche Eingangsdatum haben.

Die beiden Beispielanfragen zeigen, daß es von der Interpretation der Vorkommen von Nullwerten in einzelnen Attributen abhängen kann, ob Anfrageformulierungen zu Antworten mit gesicherter Information führen oder nicht. Vorkommen von Nullwerten, die gemäß SQL-Semantik keinerlei Information tragen sollen, kann damit durch Wahl geeigneter Anfrageformulierungen in spezifischer Weise eine Bedeutung unterlegt werden.

Wir haben schon in den Kapiteln 4 und 5 gesehen, daß die Behandlung von Duplikaten im Zusammenhang mit fehlenden Werten sorgfältig erfolgen muß, wenn Antworten mit gesicherter Information gewünscht sind. Obwohl die allgemeine Vorgehensweise bei der Definition der Semantik von SQL es nahelegt, daß gesicherte Information in Antworten auch hierbei das Ziel war, finden sich Festlegungen im Zusammenhang mit Duplikaten und ihrer Behandlung, die diesem Ziel zuwiderlaufen. Betrachten wir zunächst, wann Werte als verschieden und wann als Duplikate angesehen werden. In [ISO92] finden sich folgende Festlegungen:

Two values are said to be not distinct if either: both are the null value, or they compare equal according to Subclause 8.2, "<comparison predicate>". Otherwise they are distinct. Two rows (or partial rows) are distinct if at least one of their pairs of respective values is distinct. Otherwise they are not distinct. The result of evaluating whether or not two values or two rows are distinct is never unknown.

Two or more values or rows are said to be duplicates (of each other) if and only if they are not distinct.

Wie in den Vorschlägen von Codd ([Cod79]) und Zaniolo ([Zan84]) findet sich hier eine Festlegung, durch die Vorkommen von Nullwerten in unterschiedlichen Tupeln und/oder Relationen als identisch (Duplikate) angesehen werden. Dagegen wird dem Vergleich zweier Vorkommen von Nullwerten über ein Vergleichsprädikat für alle Vergleichsoperatoren der Wert *unbekannt* zugeordnet. In Abschnitt 4.1 haben wir uns schon einmal mit dieser Problematik befaßt.

Die Duplikatauffassung wird unter anderem bei den Mengenoperationen zugrunde gelegt. Das folgende Beispiel zeigt, daß bei der Schnittbildung damit eine (im Sinn gesicherter Information unzulässige) Identifizierung zweier Vorkommen von Nullwerten erfolgen kann.

Beispiel 8.1: Seien R und S Relationen (Basistabellen) über der Attributmenge {A, B}, in denen jeweils das Tupel (a,NULL) enthalten ist.
Für den Ausdruck

R INTERSECT S

ist dann dieses Tupel auch im Ergebnis enthalten, da seine beiden Vorkommen als Duplikate betrachtet werden.
Für den Ausdruck

SELECT x FROM R x, S y WHERE x.A = y.A AND x.B = y.B

erhalten wir dagegen die leere Antwort, da dem Vergleich NULL = NULL stets der Wert *unbekannt* zugeordnet wird.

Aus dem Beispiel ergibt sich auch, daß schon in sehr einfachen Fällen Umformungen von Anfragen gemäß üblicher Regeln (etwa in einem Optimierungsschritt) zu Änderungen von Antworten führen können. Die Duplikatregel führt zu weiteren Schwierigkeiten bei der Interpretation der Vorkommen von Nullwerten in Antworten. Während von NULL-Vorkommen in den Basistabellen des DB-Zustands davon ausgegangen wird, daß durch NULL das Fehlen genau eines Wertes modelliert wird, kann diese Annahme für NULL-Vorkommen in Antworten im allgemeinen nicht mehr gemacht werden oder die Interpretation der Tupel in Antworten selbst muß geeignet angepaßt werden.

Beispiel 8.2: Sei R = {(a,2,NULL), (b,2,NULL)} eine Relation über {A,B,C}. Die Anfrage

SELECT DISTINCT x.B, x.C FROM R x

liefert die Menge {(2,NULL)} als Ergebnis, da die beiden Vorkommen von NULL als Duplikate betrachtet werden. Damit repräsentiert das Vorkommen von NULL im Antworttupel nicht mehr die Information, daß genau ein Wert fehlt (vergleiche den Übergang von der Vervollständigung zur engen Ausdehnung bei der Interpretation von Antworten in Kapitel 4).

Zwei Werte oder Tupel sind in SQL entweder gleich oder verschieden. *Unbekannt* ist nicht möglich als Ergebnis eines Vergleichs zweier Werte oder Tupel (siehe oben). Damit können auch bei der Differenzbildung Ergebnisse erhalten werden, die keine gesicherte Information darstellen.

Beispiel 8.3: Seien R = {(a,1), (a,2), (a,3)} und S = {(a,NULL)} Relationen über der gleichen Attributmenge. Als Ergebnis des SQL-Ausdrucks R EXCEPT ALL S erhalten wir den Inhalt von R, da (a,NULL) als verschieden von allen Tupeln in R angesehen wird.

8.2 Gesicherte Antworten

Wir haben in Kapitel 3 verschiedene Formen gesicherter und möglicher Antworten auf Anfragen diskutiert und gesehen, daß zu jeder solchen Form die korrekte Bedeutung bekannt sein muß, damit aus Antworten keine falschen Schlüsse gezogen werden. In SQL können Nullwerte in Antworten wie gewöhnliche Werte vorkommen,

allerdings haben die Beispiele im vorangehenden Abschnitt gezeigt, daß die Bedeutung von Antworten von der gewählten Formulierung der Anfrage abhängig sein kann und nicht immer gesicherte Information darstellt. Ferner ist darauf zu achten, daß aufgrund der Duplikatregel Tupel und Nullwerte in Antworten eventuell eine andere Bedeutung haben als in Basistabellen.

Wir wollen im folgenden eine Definition für gesicherte Antworten auf SQL-Anfragen angeben und dabei pragmatisch vorgehen, indem wir die SQL-Semantik für diese Definition verwenden und Komplexitätsprobleme vermeiden. Außerdem nehmen wir keine Rücksicht auf das CWA-Problem, das wir im letzten Kapitel diskutiert haben.

Wegen der angesprochenen Problematik bei der Behandlung syntaktischer Duplikate, die für die Mengenoperationen wesentlich ist, beschränken wir uns im folgenden auf SQL-Anfragen ohne Mengenoperationen. Außerdem betrachten wir keine Funktionsausdrücke, da auch hier in den Standards problematische Festlegungen getroffen wurden ([Dat90]), die mit dem unten vorgestellten Verfahren nicht ohne Änderung der SQL-Semantik reparierbar sind.

In [Vas79] findet sich die Feststellung, daß es keinen Sinn macht, "nichts" mit einem Wert zu vergleichen. Mit dieser Auffassung können wir uns bei der Definition gesicherter Antworten auf eine mehr "syntaktische" Berücksichtigung von Nullwerten zurückziehen, d.h. gesicherte Antworten werden nicht mehr über mögliche DB-Zustände, sondern direkt über dem gegebenen partiellen DB-Zustand definiert. Tautologien mit Beteiligung von Nullwerten, wie wir sie bisher betrachtet haben, werden nicht mehr als Tautologien angesehen. Für die Umsetzung dieser Idee bei der Festlegung von gesicherten SQL-Antworten genügt es, sich auf das Vergleichsprädikat (<comparison predicate>) zu beziehen, da dieses Prädikat auch bei der Definition anderer Prädikate benutzt wird, wenn Werte miteinander verglichen werden. Folgende Forderung ergibt sich dann für gesicherte Antworten:

Gesicherte Antworten enthalten keine Tupel, deren Qualifikation von Werten abhängig ist, die einem Vergleichsprädikat zugeordnet werden, das Nullwerte unter seinen Argumenten hat.

Für den TRC erfüllt die ω-Auswertung diese Forderung. Die dreiwertige Logik garantiert, daß kein Tupel sich qualifiziert, wenn für den entsprechenden Qualifikationsausdruck eine Abhängigkeit von einem zu *unbekannt* ausgewerteten Vergleichsausdruck in folgender Weise besteht: Der Qualifikationsausdruck hat den Wert *wahr*, wenn dieses Vorkommen von *unbekannt* bei der Auswertung durch *wahr* oder *falsch* ersetzt wird. Diese Monotonieeigenschaft fordern wir nun auch bei der Auswertung von SQL-Anfragen und erhalten damit eine Definition für gesicherte Antworten auf SQL-Anfragen unter Verwendung der SQL-Semantik.

Definition: Ein Tupel t qualifiziert sich für die gesicherte Antwort auf eine SQL-Anfrage, falls der Qualifikationsteil der Anfrage den Wert *wahr* hat bei jeder Auswertung für t, die durch folgende Modifikation aus der SQL-Auswertung für t erhalten werden kann: Jedes Vorkommen von *unbekannt*, das als Ergebnis der Auswertung eines Prädikates erhalten wird, weil NULL als Argument auftritt, wird durch einen der beiden Booleschen Werte *wahr* oder *falsch* ersetzt.

Diese Definition erinnert an das *Nullersetzungsprinzip*, das wir zu Beginn von Abschnitt 4.1 betrachtet haben. Ein Unterschied zu diesem Prinzip besteht aber

darin, daß auch Ausdrücke der Form x.A ≥ 0 mit dom(A) = ℕ keine Tautologien darstellen.

Wir haben in Abschnitt 8.1 gezeigt, daß die SQL-Semantik trotz der Verwendung einer dreiwertigen Logik keine gesicherten Antworten im obigen Sinn garantiert. Der Grund hierfür liegt in dem Umschalten auf die Boolesche Logik bei der Auswertung von Teilanfragen unter Beschränkung auf den gesicherten Anteil der Teilantwort. In Kapitel 4 und 5 haben wir gesehen, daß das Umschalten von gesicherten auf mögliche Versionen von Operationen und umgekehrt ein Weg ist, gesicherte Antworten (im dortigen Sinn) für Algebraausdrücke zu erhalten. Da die hier betrachtete Form gesicherter Antworten offensichtlich eine Einschränkung der gesicherten uni-Antworten darstellt, bietet es sich an, die Idee des "Umschaltens" auch bei der Auswertung von SQL-Anfragen anzuwenden. Dazu definieren wir zunächst, was mögliche Antworten in Entsprechung zu den oben definierten gesicherten Antworten sind. Um den Unterschied zu den bisher verwendeten Formen möglicher Antworten deutlich zu machen (mögliche Antworten können als Antwort für einen möglichen Zustand erhalten werden), sprechen wir von *potentiellen* Antworten.

Definition: Ein Tupel t qualifiziert sich für eine potentielle Antwort auf eine SQL-Anfrage, falls der Qualifikationsteil der Anfrage den Wert *wahr* hat bei einer Auswertung für t, die durch folgende Modifikation aus der SQL-Auswertung für t erhalten werden kann: Jedes Vorkommen von *unbekannt*, das als Ergebnis der Auswertung eines Prädikates erhalten wird, weil NULL als Argument auftritt, wird durch einen der beiden Booleschen Werte *wahr* oder *falsch* ersetzt.

Diese Definition unterscheidet sich von derjenigen für gesicherte Antworten nur darin, daß die Existenz *einer* Auswertung mit der geforderten Eigenschaft genügt. Der Unterschied zu möglichen Antworten ergibt sich hier entsprechend wie bei gesicherten Antworten dadurch, daß bestimmte Tautologien (und Widersprüche) nicht gültig sind.

Mit der bekannten Äquivalenz

$$(\forall x)(\Phi(x)) \equiv \neg\,(\exists x)(\neg\,\Phi(x))$$

aus der Prädikatenlogik läßt sich zeigen, daß das durch eine Negation bewirkte "Umschalten" von gesicherten auf potentielle Antworten und umgekehrt zu korrekten Ergebnissen führt:
Die gegenseitige Unabhängigkeit der Ersetzungen von Vorkommen von *unbekannt* bei den zu betrachtenden Auswertungen bewirkt, daß alle Prädikate in WHERE-Klauseln von Anfragen unabhängig voneinander betrachtet werden können. Nehmen wir an, daß die Variable x für eine Auswertung steht und p(x) für den Wert eines solchen Prädikates p mit einer Auswertung x, dann ergibt sich die Äquivalenz

$$(\forall x)(\text{NOT } p(x)) \equiv \text{NOT } (\exists x)(p(x)).$$

Äquivalenzen aus der Prädikatenlogik dürfen verwendet werden, da bei den betrachteten Auswertungen nur Boolesche Werte auftreten. Wir erhalten also ein gesichertes Ergebnis, wenn wir für das Prädikat p so verfahren, als sei eine potentielle Antwort gewünscht.

Potentielle Antworten erleichtern die Angabe des Verfahrens zur Umformung von SQL-Anfragen, das wir im nächsten Abschnitt vorstellen. Davon abgesehen haben sie auch allein betrachtet eine Bedeutung als sinnvolle Antwortform, insbe-

sondere weil sie effizient berechnet werden können. Zu beachten ist bei ihrer Verwendung, daß in Antworten Tupel auftreten können, die für keinen möglichen Zustand, wie wir sie für die "no information"-Interpretation von Nullwerten betrachtet haben, in der Antwort enthalten sind.

8.3 Systematische Umformung von SQL-Anfragen

Die Bedeutung einer SELECT .... FROM .... WHERE .... -Klausel ist wie folgt festgelegt: Zunächst wird das Kreuzprodukt der in der FROM-Klausel angegebenen Relationen gebildet; anschließend wird die in der WHERE-Klausel enthaltene Bedingung für alle Tupel der durch die Kreuzproduktbildung erhaltenen Relation überprüft. Dabei wird die dreiwertige Logik angewandt, und es qualifizieren sich nur solche Tupel, für die die Überprüfung der Bedingung *wahr* ergibt. Dies bedeutet, daß ein Ergebnis im Sinne der oben definierten gesicherten Antwort erhalten wird unter der Voraussetzung, daß in die Auswertung der Bedingung nur gesicherte Information einfließt. Auf dieses Ergebnis wird im letzten Schritt eine Projektionsoperation gemäß der SELECT folgenden Selektionsliste angewandt.

Tritt wie in unserer ersten SQL-Anfrage in Abschnitt 8.1 eine Negation in einer WHERE-Klausel einer SELECT-Anweisung auf, dann muß der negierte Teilausdruck insgesamt gesicherte Information darstellen, damit sich nur gesicherte Tupel für das Ergebnis der WHERE-Klausel qualifizieren. Dies bedeutet, daß für den Teilausdruck - in geeigneter Weise als Anfrage betrachtet - die potentielle Antwort bestimmt werden muß. Für das EXISTS-Prädikat

```
EXISTS
 (SELECT *
  FROM PERSON y
  WHERE x.NACHNAME = y.NACHNAME
     AND x.VORNAME <> y.VORNAME
     AND x.GEBURTSTAG = y.GEBURTSTAG)
```

aus der Beispielanfrage heißt das, daß auch alle Tupel aus der Relation PERSON in das Ergebnis der Auswertung der WHERE-Klausel mit aufgenommen werden müssen, für die *unbekannt* als Wert erhalten wird. In diesen Fällen ist allgemein eine Ersetzung der Vorkommen von *unbekannt* durch *wahr* und/oder *falsch* möglich, so daß der gesamte Ausdruck den Wert *wahr* hat. Dies folgt aus der angenommenen Unabhängigkeit der Ersetzung aller Vorkommen von *unbekannt*. Ohne diese Voraussetzung könnte der Schluß nicht gezogen werden.

In SQL steht mit dem IS NULL-Prädikat eine Möglichkeit zur Verfügung, für jeden beliebigen Attributwert festzustellen, ob es sich bei ihm um den Nullwert handelt oder nicht. Mit Hilfe dieses Booleschen Prädikates kann somit das Vorkommen von *unbekannt* bei der Auswertung von Vergleichsausdrücken in einer WHERE-Klausel überprüft und eine entsprechende "Steuerung" des Ergebnisses der WHERE-Klausel vorgenommen werden. In unserer Beispielfrage genügt es, OR y.GEBURTSTAG IS NULL der WHERE-Klausel in der Teilanfrage hinzuzufügen, um den gewünschten Effekt, eine potentielle Antwort, zu erzielen. Analog können durch derartige Modifikationen von Ausdrücken gesicherte Antworten erhalten werden. Weiterhin kann erreicht werden, daß *unbekannt* nicht als Ergebnis einer Prädikatauswertung auftritt. Wie eine solche Steuerung systematisch für alle Prädikate erfolgen kann, werden wir im folgenden zeigen. Zusammen mit der Anwendung der switch-Technik,

wie wir sie schon in Kapitel 4 und 5 angewandt haben, erhalten wir dann ein Verfahren zum Umformen von SQL-Anfragen in Anfragen, die mit der Standard-SQL-Semantik stets gesicherte Antworten gemäß der oben gegebenen Definition liefern.

Zu jedem Prädikat in SQL geben wir an, wie eine gesicherte und wie eine potentielle Version erhalten werden kann und welche Version für geschachtelt auftretende Teilanfragen gewählt werden muß. Dabei orientieren wir uns am syntaktischen Aufbau der Prädikate. Zur Vereinfachung geben wir immer nur für ein festes Attribut A oder B aus einer Liste von betroffenen Attributen den zu erzeugenden Ausdruck an und deuten durch .... an, daß bei mehrelementigen Listen die Ergänzung durch entsprechende Ausdrücke sinngemäß zu erfolgen hat. Attribute können nur dann betroffen sein, wenn für sie das Vorkommen von Nullwerten im DB-Schema nicht ausgeschlossen wird.

Comparison predicate (wir nehmen für dieses Prädikat Bezug auf die etwas einfachere Syntax von [ISO89]; es geht dadurch inhaltlich gegenüber [ISO92] nichts Wesentliches verloren)

<value expression> <compop> {<value expression> | <subquery>}

1) <value expression> <compop> <value expression>

a) gesicherte Version:
Einem Ausdruck v Θ w wird der Wert *unbekannt* zugeordnet, falls NULL für irgendein Attribut, das in v oder w vorkommt, als Wert eingesetzt wird, d.h. falls das für eine entsprechende Variable eingesetzte Tupel in dem Attribut den Wert NULL hat. Der Ausdruck ist daher zu ergänzen um

AND A IS NOT NULL ....

für alle betroffenen Attribute A aus v und w. Dadurch wird immer *falsch* statt *unbekannt* erhalten, wenn eines der Argumente in den Wertausdrücken der Nullwert ist.

b) potentielle Version:
Der Ausdruck ist zu ergänzen um

OR A IS NULL ....

für alle betroffenen Attribute A aus v und w. Auf die Notwendigkeit einer eventuellen Klammerung in Abhängigkeit von der Umgebung, in der der Ausdruck auftritt, ist hierbei zu achten. Durch diese Ergänzung wird *wahr* statt *unbekannt* erhalten, wenn eines der Argumente in den Wertausdrücken der Nullwert ist.

2) <value expression> <compop> <subquery>

a) gesicherte Version:
Für den Ausdruck v Θ (S), S Teilanfrage, muß zu S die gesicherte Version erzeugt werden. Der Ausdruck hat nach SQL-Semantik den Wert

undefiniert, falls |S| > 1

unbekannt, falls |S| = 0 oder
v ≡ NULL oder
S = {(NULL)}

Der Ausdruck ist zu ergänzen um

AND A IS NOT NULL .... AND EXISTS (S')

für alle betroffenen Attribute A aus v. Ist S eine Teilanfrage der Form

SELECT w
FROM ....
WHERE Φ

dann hat S' die Form

SELECT w
FROM ....
WHERE Φ AND NOT (B IS NULL OR ....)

wobei in der Disjunktion B IS NULL OR .... alle betroffenen Attribute aus w auftreten müssen. Mit dieser Ergänzung wird erreicht, daß in allen drei Fällen, in denen das Prädikat den Wert *unbekannt* liefert, *falsch* als Wert erhalten wird. Die Modifikation von S ist dabei notwendig, um den dritten Fall, das Tupel (NULL), als Ergebnis der Teilanfrage auszuschließen. Auf ein Problem mit der Festlegung im ersten Fall, |S| = 0, kommen wir gleich zurück. Im Fall |S| > 1 bewirkt das Ergebnis *undefiniert*, daß die SQL-Anweisung mit einer Fehlermeldung endet.

b) potentielle Version:
Auch für S ist die potentielle Version zu bilden. Statt *unbekannt* ist *wahr* zu erzeugen außer im Fall |S| = 0. In diesem Fall liegt nicht notwendigerweise eine Abhängigkeit von einem Nullwert vor, weshalb wir ihn getrennt diskutieren (siehe Bemerkung unten). Wir erhalten den gewünschten Effekt, wenn wir v Θ (S) um

(OR A IS NULL ....) AND EXISTS (S) OR EXISTS (S')

ergänzen (unter Beachtung eventuell notwendiger Klammersetzung). Dabei ergibt sich S' in diesem Fall aus S wie folgt:

SELECT w
FROM ....
WHERE Φ AND (A IS NULL OR ....)

Bemerkung: Mit der Festlegung, daß c Θ Φ den Wert *unbekannt* hat für jede Konstante c einschließlich des Nullwertes, ergibt sich ein Problem mit der Semantik von SQL, das unabhängig von Nullwerten ist und daher auch nicht durch die hier betrachteten Umformungsregeln gelöst werden kann. Wir lassen es daher offen. Das folgende Beispiel macht das Problem deutlich (wir greifen hier auf die schon bekannte Relation zurück, auch wenn die Anfrage nicht ganz sinnvoll ist).
Sei in der Relation PERSON aus Abschnitt 8.1 der Nullwert durch den Wert '19.1.' ersetzt:

PERSON

NACHNAME	VORNAME	GEBURTSTAG
Schmidt	Richard	19.1.
Schmidt	Willi	19.1.

Betrachte folgende Anfrage:

"Gib die Daten der Personen, deren Name verschieden ist von dem Namen der am 28.2. geborenen Personen."

Mit der Formulierung

```
SELECT *
FROM PERSON x
WHERE x.NAME <> (SELECT y.NAME
                 FROM PERSON y
                 WHERE y.GEBURTSTAG = 28.2.)
```

erhalten wir die leere Antwort, da kein Tupel sich für die Teilanfrage qualifiziert und der Vergleich mit der leeren Menge jeweils zu *unbekannt* ausgewertet wird. Die Auswertung kann so gesehen werden, als ob sie unter der open world assumption erfolgt.

Die beiden Tupel aus der Relation PERSON erhalten wir dagegen mit folgender Formulierung als Antwort:

```
SELECT *
FROM PERSON x
WHERE NOT EXISTS (SELECT *
                  FROM PERSON y
                  WHERE y.NAME = x.NAME
                    AND  y.GEBURTSTAG = 28.2.)
```

In predicate

<value expression> [NOT] IN {<subquery> | (<in value list>)}

Für dieses Prädikat sind folgende Äquivalenzen festgelegt:

x IN S ≡ x = SOME S und x NOT IN S ≡ NOT(x IN S)

Es muß daher nicht weiter betrachtet werden.

Quantified comparison predicate

<row value constructor> <compop> <quantifier> <table subquery>

Zur Vereinfachung sei angenommen, daß <row value constructor> aus einem einfachen Attribut A besteht. Es ist eine Existenz-Quantifizierung (SOME oder ANY) und eine universelle Quantifizierung (ALL) einer Teilanfrage S möglich, wobei für <compop> die üblichen Vergleichsoperatoren eingesetzt werden können. Stehe Θ im folgenden wieder für einen solchen Vergleichsoperator.

1) A Θ SOME(S)

a) gesicherte Version:
Für S ist ebenfalls die gesicherte Version einzusetzen. Es sind zwei kritische Fälle zu betrachten, in denen Ergänzungen vorzunehmen sind, falls im Attribut A bzw. B Nullwerte erlaubt sind:

i) A ≡ NULL.
NULL Θ SOME(S) erhält den Wert *falsch*, falls die Auswertung von S ein leeres Resultat ergibt, und *unbekannt*, sonst. Anstelle von *unbekannt* muß *falsch* erzeugt werden, um möglicherweise folgende Negationen korrekt wirken lassen zu können. Dies können wir durch Hinzufügen von

```
AND A IS NOT NULL
```

zum Prädikat sicherstellen.

ii) In der Antwort auf S ist NULL enthalten.
Betrachte als Beispiel den Ausdruck 5 >= SOME({7,8,NULL}). Gemäß der SQL-Semantik erhält dieser Ausdruck den Wert *unbekannt*. Um eine Abhängigkeit vom Vorkommen des Nullwertes im Ergebnis der Auswertung von S auszuschließen, genügt es, ihn aus dem Ergebnis herauszuhalten. Dies kann durch eine entsprechende Ergänzung der WHERE-Klausel der Teilanfrage erreicht werden:

SELECT B		SELECT B
FROM	$\longrightarrow$	FROM ....
WHERE Φ		WHERE Φ AND B IS NOT NULL

b) Potentielle Version:
In diesem Fall ist für S die potentielle Antwort sicherzustellen. Es sind die gleichen kritischen Fälle wie bei der gesicherten Version zu betrachten.

i) A $\equiv$ NULL.
Falls das Ergebnis der Teilanfrage S nicht leer ist, muß dem Ausdruck der Wert *wahr* statt *unbekannt* zugewiesen werden. Um dies zu bewirken, wird dem Prädikat

OR A IS NULL AND EXISTS(S)

hinzugefügt, falls A ein Attribut ist, in dem Nullwerte vorkommen können. Für S in EXISTS(S) ist dabei ebenfalls die potentielle Version erforderlich.

ii) In der Antwort auf S ist NULL enthalten.
Betrachte wieder das Beispiel 5 >= SOME({7,8,NULL}). Der Wert des Ausdrucks ist gleich dem Wert des Vergleichs 5 >= NULL, da die anderen Vergleiche *falsch* ergeben. Um in solchen Situationen *wahr* als Wert erhalten zu können, ergänzen wir das Prädikat um

OR EXISTS(S'),

wobei S' sich aus S (S wie oben) durch Hinzufügen von

AND B IS NULL

zum Ausdruck Φ der WHERE-Klausel ergibt. Dabei ist für S' ebenfalls wieder die potentielle Version zu erzeugen. Die Ergänzung kann unterbleiben, falls in B keine Nullwerte erlaubt sind.

2) A Θ ALL(S)

a) Gesicherte Version:
In diesem Fall muß für S die potentielle Version erzeugt werden, um sicherzustellen, daß durch keine Möglichkeit zur Ersetzung eines Vorkommens von *unbekannt* bei der Auswertung von S sich ein Tupel qualifizieren kann, das für den Vergleich mit dem Wert in A *falsch* liefert, wodurch das gesamte Prädikat den Wert *falsch* erhält. Wir haben wieder die beiden kritischen Fälle von oben zu betrachten.

i) A $\equiv$ NULL.
NULL Θ ALL(S) erhält nach SQL-Semantik den Wert *wahr*, falls die Auswertung von S ein leeres Resultat liefert, und den Wert *unbekannt* in allen anderen Fällen. Da wir an der gesicherten Version interessiert sind, ist *falsch* an Stelle von *unbekannt* als Ergebnis zu erzeugen. Wir können dies durch Hinzufügen von

AND (A IS NOT NULL OR NOT EXISTS(S))

zum Prädikat bewirken. Dabei muß für S in NOT EXISTS(S) die potentielle Version gewählt werden. Die Ergänzung kann vollständig unterbleiben, falls in A keine Nullwerte erlaubt sind. Der Teil OR NOT EXISTS(S) ist nur notwendig, um die Zuordnung von *wahr* im Fall einer leeren Antwort auf die Teilanfrage S zu gewährleisten.

ii) Die Antwort auf S enthält den Wert NULL.
Betrachte hier als Beispiel 5 <= ALL({7,8,NULL}). Als Ergebnis muß *falsch* erhalten werden. Um dies allgemein in solchen Fällen zu erreichen, fügen wir dem Prädikat

AND NOT EXISTS(S')

hinzu, wobei S' aus S durch Anhängen von

AND B IS NULL

an Φ entsteht. Für S' ist wiederum die potentielle Version erforderlich. Die Ergänzung ist auch hier nur notwendig, falls in B Nullwerte erlaubt sind. Ist dies nicht der Fall, hat S' immer eine leere Antwort und der hinzugefügte Ausdruck ist immer *wahr*.

b) Potentielle Version:
Analog zur gesicherten Version ergibt sich hier eine Umkehrung der für S zu bestimmenden Version, d.h. für S ist die gesicherte Version notwendig. Es sind wieder die beiden kritischen Fälle zu betrachten. Dabei gelten die Regeln, wann Ergänzungen überflüssig sind, sinngemäß.

i) A $\equiv$ NULL.
An Stelle von *unbekannt* muß *wahr* als Wert erhalten werden. Dazu genügt es, dem Prädikat

OR A IS NULL

hinzuzufügen. AND EXISTS(S) muß nicht ergänzt werden wie im Fall 1)b)i), da c Θ ALL(S) den Wert *wahr* erhält für jeden Wert c, falls die Antwort auf S leer ist.

ii) Die Antwort auf S enthält den Wert NULL.
In diesem Fall kann wie im Fall 1)a)ii) vorgegangen werden.

Bemerkung: Betrachte die Formulierung unserer Beispielanfrage mit dieser Prädikatform aus Abschnitt 8.1:

```
SELECT *
FROM PERSON x
WHERE x.GEBURTSTAG <> ALL
                (SELECT y.GEBURTSTAG
                 FROM PERSON y
                 WHERE x.NACHNAME = y.NACHNAME
                    AND x.VORNAME <> y.VORNAME)
```

Diese Formulierung liefert eine gesicherte Antwort, da die gesicherte und die potentielle Antwort für die Teilanfrage übereinstimmen und *unbekannt* als Ergebnis der äußeren WHERE-Klausel wie *falsch* wirkt.

Exists predicate

EXISTS <table subquery>

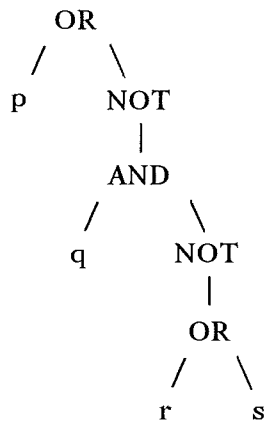
Es ist keine Umformung vorzunehmen, sondern lediglich zu beachten, daß für die Teilanfrage die gesicherte bzw. potentielle Version zu wählen ist, wenn für das Prädikat die gesicherte bzw. potentielle Version benötigt wird.

Die übrigen in [ISO92] aufgeführten Prädikate können entweder wie beim Vergleichsprädikat im Fall <value expression> <compop> <value expression> behandelt werden (like-, between predicate) oder sie haben stets ein Resultat ungleich *unbekannt* (unique-, match-, overlaps predicate).

Ob die gesicherte oder die potentielle Version für ein Prädikat erzeugt werden muß, ergibt sich aus der Struktur der betrachteten Anfrage und den Vorschriften, die in den angegebenen Regeln enthalten sind. Wie bei der switch-Auswertung in Abschnitt 4.4 muß bei Auftreten einer Negation von gesichert auf potentiell und umgekehrt "umgeschaltet" werden, d.h. für alle Prädikate auf äußerem Niveau des negierten Teilausdrucks ist die jeweils andere Version zu wählen. Betrachte etwa den folgenden Ausdruck mit vier beliebigen Prädikaten p,q,r und s:

p OR NOT(q AND NOT(r OR s))

Soll für den gesamten Ausdruck die gesicherte Version bestimmt werden, dann muß diese Version auch für p, r und s gewählt werden. Für q ist dagegen die potentielle Version notwendig. Die Bestimmung der Versionen kann auf einfache Weise über den Strukturbaum zum Ausdruck erfolgen, den man erhält, indem man die Prädikate als Blätter und die Booleschen Operatoren als innere Knoten hernimmt (vergleiche die Operatorbäume, die wir in Kapitel 4 bei der switch-Auswertung für die Relationenalgebra betrachtet haben, siehe etwa Beispiel 4.17, S. 90). Für den Beispielausdruck ergibt sich:



Sind wir an gesicherten Antworten interessiert, muß für die Bestimmung der Versionen an der Wurzel des Operatorbaumes zur äußeren WHERE-Klausel mit gesicherten Versionen gestartet werden. Entsprechend können potentielle Antworten auf Anfragen erhalten werden, wenn bei der Umformung mit potentiellen Versionen gestartet wird. Für die WHERE-Klauseln in Teilanfragen können ebenfalls solche Bäume erzeugt werden. Die Startversion ergibt sich dann aus der Version des Prädikats, in dem die Teilanfrage auftritt.

Beispiel 8.4: Wir geben für die zweite Formulierung unseres Geburtstagsbeispiels die ergänzte Formulierung an, die sich durch Anwendung des Verfahrens ergibt.

Wir erhalten nach der Ergänzung eine Formulierung, die mit der SQL-Semantik eine leere Antwort für den in Abschnitt 8.1, S. 172, angegebenen Zustand liefert, was der gesicherten Antwort entspricht.

```
SELECT *
FROM PERSON x
WHERE x.NACHNAME <> ALL
                (SELECT y.NACHNAME
                 FROM PERSON y
                 WHERE (x.GEBURTSTAG = y.GEBURTSTAG
                        OR y.GEBURTSTAG IS NULL
                        OR x.GEBURTSTAG IS NULL)
                   AND x.VORNAME <> y.VORNAME)
```

Eine einfache Überprüfung aller angegebenen Umformungsregeln zeigt, daß die erzeugten Versionen, geeignet als Anfragen aufgefaßt, stets die Antwort in der geforderten Form garantieren. Im Fall totaler DB-Zustände verhalten sich alle hinzugefügten Ausdrücke neutral, d.h. sie haben immer den Wert *wahr*, falls sie mit AND angefügt werden, und immer den Wert *falsch,* falls die Ergänzung mit OR erfolgt. Als einzige Ausnahme haben wir den Fall der leeren Antwort auf eine Teilanfrage diskutiert, der wegen der aufgezeigten Probleme ausgeklammert werden muß. Mit der in Abschnitt 8.2 gezeigten Korrektheit der "Umschaltung" von gesicherten auf potentielle Antworten und umgekehrt ergibt sich damit:

Satz 8.1: Zu jeder SQL-Anfrage q liefert das beschriebene Verfahren eine modifizierte Anfrage q', die im Fall totaler DB-Zustände die gleiche SQL-Antwort wie q hat und im Fall von Nullwerten in Basistabellen eines DB-Zustandes eine gesicherte oder potentielle Antwort als SQL-Antwort ergibt. Welche der beiden Versionen erhalten wird, ist durch die Festlegung der Version für die äußere SELECT-Anweisung von q bestimmt.

Für die Bestimmung einer gesicherten Antwort ist es gemäß SQL-Semantik gleich, ob der Wert einer WHERE-Klausel *falsch* oder *unbekannt* ist. Wir haben das Erzwingen von *falsch* statt *unbekannt* in den gesicherten Versionen der Prädikate nur benötigt, um eventuelle Negationen einfach und korrekt behandeln zu können. Zur Vereinfachung der durch das Verfahren erzeugten Anfragen können vorgenommene Ergänzungen von Prädikaten wieder rückgängig gemacht werden, wenn folgendes gilt:
Für ein Prädikat p ist die gesicherte Version notwendig, p ist aber nicht direkt in einem negierten Teilausdruck einer WHERE-Klausel enthalten.
In solchen Fällen kann *unbekannt* durch *falsch* und umgekehrt *falsch* durch *unbekannt* ersetzt werden, ohne daß sich der Wert des Prädikates ändert.

Gemäß diesen Möglichkeiten können in der SQL-Formulierung des folgenden Beispiels alle Ergänzungen wegfallen, ohne daß sich an der Antwort etwas ändert. Wie in Abschnitt 8.1 erwähnt, erhält man schon für die ursprüngliche Formulierung Antworten mit gesicherter Information für die dort betrachtete Interpretation fehlender Attributwerte.

Beispiel 8.5: Die strikte Anwendung des Verfahrens auf die dritte Formulierung unserer Beispielanfrage in Abschnitt 8.1 liefert folgende Formulierung, in der wir die Ergänzungen kursiv abgehoben haben:

```
SELECT *
FROM PERSON x
WHERE x.GEBURTSTAG <> ALL
                (SELECT y.GEBURTSTAG
                 FROM PERSON y
                 WHERE x.NACHNAME = y.NACHNAME
                     AND x.VORNAME <> y.VORNAME)
          AND (x.GEBURTSTAG IS NOT NULL  OR
                NOT EXISTS  (SELECT z.GEBURTSTAG
                             FROM PERSON z
                             WHERE x.NACHNAME = z.NACHNAME
                                 AND x.VORNAME <> z.VORNAME))
          AND NOT EXISTS (SELECT y.GEBURTSTAG
                          FROM PERSON y
                          WHERE x.NAME = y.NAME
                              AND x.VORNAME <> y.VORNAME
                              AND y.GEBURTSTAG IS NULL)
```

Literaturverzeichnis

[AG85] Abiteboul S. und Grahne G.: Update semantics for incomplete databases. *Proc. VLDB 85,* Stockholm, S. 1-12, 1985

[AKG87] Abiteboul S., Kanellakis P. und Grahne G.: On the representation and querying of sets of possible worlds. *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, San Francisco, S. 34-48, 1987

[AKG91] Abiteboul S., Kanellakis P. und Grahne G.: On the representation and querying of sets of possible worlds. *Theoretical Computer Science* 78, S. 159-187, 1991

[ANSI75] ANSI/X3/SPARC: Study Group on Data Base Management Systems: Interim Report. *FDT (Bulletin of ACM SIGMOD)* 7 (2), S. 3-140, 1975

[BB92] Bolc L. und Borowik P.: *Many-Valued Logics, 1 Theoretical Foundations.* Springer-Verlag, Berlin, 1992

[Bel77] Belnap N.D.: A useful four-valued logic. In: *Modern Uses of Multiple-Valued Logic* (Hrg.: J.M. Dunn und G. Epstein), D. Reidel, Dordrecht, S. 8-37, 1977

[Bis81] Biskup J.: A formal approach to null values in database relations. In: *Advances in Data Base Theory* 1 (Hrg.: H. Gallaire, J. Minker und J.-M. Nicolas), Plenum Press, S. 299-341, 1981

[Bis83] Biskup J.: A foundation of Codd's relational maybe-operations. *ACM Trans. on Database Systems* 8 (4), S. 608-636, 1983

[Bis84] Biskup J.: Extending the relational algebra for relations with maybe tuples and existential and universal null values. *Fundamenta Informaticæ* VII.1, S. 129-150, 1984

[BM75] Barnes D.W. und Mack J.M.: *An Algebraic Introduction to Mathematical Logic.* Springer-Verlag, Berlin, 1975

[Bru89] Brudno V.A.: Valuations in incomplete information databases. *Information Sciences* 47, S. 389-398, 1989

[BS76] Belnap N.D. und Steel T.B.: *The Logic of Questions and Answers.* Yale University Press, London, 1976

[Cod70] Codd E.F.: A relational model of data for large shared data banks. *Comm. ACM* 13 (6), S. 377-387, 1970

[Cod72a] Codd E.F.: Further normalization of the database relational model. In: *Data Base Systems. Current Computer Science Symposium* 6 (Hrg.: R. Rustin), Prentice Hall, Englewood Cliffs, S. 33-64, 1972

[Cod72b] Codd E.F.: Relational completeness of data base sublanguages. In: *Data Base Systems. Current Computer Science Symposium* 6 (Hrg.: R. Rustin), Prentice Hall, Englewood Cliffs, S. 65-98, 1972

[Cod75] Codd E.F.: Understanding relations (Installment No. 7). *FDT (Bulletin of ACM SIGMOD)* 7 (3-4), S. 23-28, 1975

[Cod79] Codd E.F.: Extending the database relational model to capture more meaning. *ACM Trans. on Database Systems* 4 (4), S. 397-434, 1979

[Cod86] Codd E.F.: Missing information (applicable and inapplicable) in relational databases. *ACM SIGMOD RECORD* 15 (4), S. 53-77, 1986

[Cod87] Codd E.F.: More commentary on missing information in relational databases (applicable and inapplicable information). *ACM SIGMOD RECORD* 16 (1), S. 42-45, 1987

[Cod90] Codd E.F.: Missing information. In: *The Relational Model for Database Management: Version 2*, Addison-Wesley, Reading, Kapitel 8 u. 9, 1990

[Dat86] Date C.J.: Null values in database management. In: *Relational Database: Selected Writings*. Addison-Wesley, Reading, Kapitel 15, 1986

[Dat90] Date C.J.: *Relational Database Writings 1985-1989*. Addison-Wesley, Reading, 1990

[DT91] Delahaye J.P. und Thibau V.: Programming in three-valued logic. *Theoretical Computer Science* 78, S. 189-216, 1991

[EFT92] Ebbinghaus H.-D., Flum J. und Thomas W.: *Einführung in die mathematische Logik*. 3. Auflage, BI-Verlag, Mannheim, 1992

[EN94] El Masri R. und Navathe S.: *Fundamentals of Database Systems*. 2. Auflage, Benjamin/Cummings, Redwood City, 1994

[Fit69] Fitting M.C.: *Intuitionistic Logic Model Theory and Forcing*. Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam, 1969

[Fit91] Fitting M.C.: Kleene's logic, generalized. *J. Logic Computation* 1 (6), S. 797-810, 1991

[GaMi78] Gallaire H. und Minker J.: *Logic and Data Bases*. Plenum Press, New York, 1978

[GaMo90] Garcia O.N. und Moussavi M.: A six-valued logic for representing incomplete knowledge. *Proc. IEEE 20th Int. Symp. on Multiple-valued Logic* (Hrg.: G. Epstein), Charlotte, S. 110-114, 1990

[Ges91] Gessert G.H.: Handling missing data by using stored truth values. *ACM SIGMOD RECORD* 20 (3), S. 30-42, 1991

[Ges90] Gessert G.H.: Four valued logic for relational database systems. *ACM SIGMOD RECORD* 19 (1), S. 29-35, 1990

[GJ90] Garey M.R. und Johnson: *Computers and Intractability*. Freeman, New York, 1990

[Gld81] Goldstein B.S.: Constraints on null values in relational databases. *Proc. 7th VLDB Conf.*, Cannes, S. 101-110, 1981

[Gls85] Golshani F.: Growing certainty with null values. *Information Systems* 10 (3), S. 289-297, 1985

[GMN84] Gallaire H., Minker N. und Nicolas J.M.: Logic and databases: a deductive approach. *ACM Computing Surveys* 16 (2), S. 153-185, 1984

[Gra84] Grahne G.: Dependency satisfaction in databases with incomplete information. *Proc. 10th VLDB Conf.*, Singapore, S. 37-45, 1984

[Gra91] Grahne G.: *The Problem of Incomplete Information in Relational Databases*. LNCS 554, Springer-Verlag, Berlin, 1991

[Grt77] Grant J.: Null values in relational data base. *Inform. Proc. Letters* 6 (5), S. 156-157, 1977

[Grt79] Grant J.: Partial values in a tabular database model. *Inform. Proc. Letters* 9 (2), S. 97-99, 1979

[Grt80] Grant J.: Incomplete information in a relational database. *Fundamenta Informaticæ* III.3, S. 363-378, 1980

[GZ88] Gottlob G. und Zicari R.: Closed world databases opened through null values. *Proc. 14th VLDB Conf.*, Los Angeles, S. 50-61, 1988

[IL81] Imielinski T. und Lipski W.: On representing incomplete information in a relational data base. *Proc. 7th VLDB Conf.*, Cannes, S. 388-397, 1981

[IL84] Imielinski T. und Lipski W.: Incomplete information in relational databases. *J. of the ACM* 31 (4), S. 761-791, 1984

[Imi91] Imielinski T.: Abstraction in query processing. *J. of the ACM* 38 (3), S. 534-558, 1991

[INV91] Imielinski T., Naqvi S. und Vadaparty K.: Querying design and planning databases. *Proc. Int. Conf. on Deductive and Object-Oriented Databases*, (Hrg.: C. Delobel, M. Kifer und Y. Masunage), München, LNCS 566, S. 524-545, Springer-Verlag, Berlin, 1991

[ISO89] *International Standard ISO/IEC 9075.* Second edition, Genf, 1989

[ISO92] *International Standard ISO/IEC 9075.* Third edition, Genf, 1992

[IV89] Imielinski T. und Vadaparty K.: Complexity of query processing in databases with or-objects. *Proc. 8th ACM Symp. on Principles of Database Systems*, Philadelphia, S. 51-65, 1989

[Jae78] Jaegermann M.: Information storage and retrieval systems with incomplete information. *Fundamenta Informaticæ* 2, S. 17-41, 1978

[Kel86] Keller A.M.: Set-theoretic problems of null completion in relational databases. *Inform. Proc. Letters* 22, S. 261-265, 1986

[KK93] Kandzia P. und Klein H.-J.: *Theoretische Grundlagen relationaler Datenbanksysteme.* Reihe Informatik, Band 79, BI-Verlag, Mannheim, 1993

[Kle38] Kleene S.C.: On a notation of ordinal numbers. *The Journal of Symbolic Logic* 3, S. 150-155, 1938

[Kle52] Kleene S.C.: *Introduction to Meta-Mathematics.* North-Holland, 1952

[Kln94] Klein H.-J.: How to modify SQL queries in order to guarantee sure answers. *ACM SIGMOD RECORD* 23 (3), S. 14-20, 1994

[KS95] Kimball R. und Strehio K.: Why decision support fails and how to fix it. *ACM SIGMOD RECORD* 24 (3), S. 92-97, 1995

[KW84] Keller A.M. und Winslett Wilkins M.W.: Approaches for updating databases with incomplete information and nulls. *Proc. IEEE Int. Conf. on Data Engineering*, Los Angeles, S. 332-340, 1984

[KW85] Keller A.M. und Winslett Wilkins M.W.: On the use of an extended relational model to handle changing incomplete information. *IEEE Transactions on Software Engineering*, SE-11, 7, S. 620-633, 1985

[Lak89] Lakshmanan V.S.: Query evaluation with null values: how complex is completeness?. *Proc. 9th Conf. on Found. of Software Technology and Theoretical Comp. Science*, Bangalore, LNCS 405, Springer-Verlag, Berlin, S. 204-222, 1989

[LeLi86] Lerat N. und Lipski J.: Nonapplicable nulls. *Theoretical Computer Science* 46, S. 67-82, 1986

[LeLo93] Levene M. und Loizou G.: A fully precise null extended nested relational algebra. *Fundamenta Informaticæ* 19, S. 303-343, 1993

[Ler86] Lerat N.: Query processing in incomplete logical databases. *Proc. Int. Conf on Database Theory* (Hrg.: G. Ausiello und P. Atzeni), Rome, LNCS 243, Springer-Verlag, Berlin, S. 260-277, 1986

[Lib95a] Libkin L.: Approximation in databases. *Proc. Int. Conf. on Database Theory* (Hrg.: G. Gottlob und M.Y. Vardi), Prag, LNCS 893, Springer-Verlag, Berlin, S. 411-424, 1995

[Lib95b] Libkin L.: A semantics-based approach to design of query languages for partial information. *Proc. Workshop Semantics in Databases* (Hrg.: B. Thalheim), Prag, TU Cottbus, S. 63-80, 1995

[Lie79] Lien Y.: Multivalued dependencies with null values in relational data bases. *Proc. 5th VLDB Conf.*, Rio de Janeiro, S. 61-66, 1979

[Lie81] Lien Y.: Hierarchical schemata for relational database schemata. *ACM Trans. on Database Systems* 6 (1), S. 48-69, 1981

[Lip76] Lipski W.: Informational systems with incomplete information. *Proc. 3rd Int. Symp. on Automata, Languages and Programming* (Hrg.: Michaelson S. und R. Milner), Edinburgh, S. 120-130, 1976

[Lip79] Lipski W.: On semantic issues connected with incomplete information databases. *ACM Trans. on Database Systems* 4 (3), S. 262-296, 1979

[Lip81] Lipski W.: On databases with incomplete information. *J. of the ACM* 18 (1), S. 41-70, 1981

[Lip84] Lipski W.: On relational algebra with marked nulls. *Proc. 3rd ACM Symp. on Principles of Database Systems*, S. 201-203, 1984

[LS90] Liu K.C. und Sunderraman R.: Indefinite and maybe information in relational databases. *ACM Trans. on Database Systems* 15 (1), S. 1-39, 1990

[LS91] Liu K.C. und Sunderraman R.: A generalized relational model for indefinite and maybe information. *IEEE Trans. on Knowledge and Data Engineering* 3 (1), S. 65-76, 1991

[LZ91] Liu K.C. und Zhang L.: Natural joins in relational databases with indefinite and maybe information. *Proc. 7th IEEE Int. Conf. on Data Engineering*, S. 132-139, 1991

[Mai83] Maier D.: *The Theory of Relational Databases.* Computer Science Press, Rockville, 1983

[Min82] Minker J.: On indefinite databases and the closed world assumption. *Proc. 6th Conf. on Automated Deduction*, New York, LNCS 138, Springer-Verlag, Berlin, S. 292-308, 1982

[Mor90] Morrissey J.M.: Imprecise information and uncertainty in information systems. *ACM Trans. on Information Systems* 8 (2), S. 159-180, 1990

[Mot89] Motro A.: Integrity = validity + completeness. *ACM Trans. on Database Systems* 14 (4), S. 480-502, 1989

[Mot94] Motro A.: Intensional answers to database queries. *IEEE Trans. on Knowledge and Data Engineering* 6 (3), S. 444-454, 1994

[MS93] Melton J. und Simon A.R.: *Understanding the New SQL: A Complete Guide.* Morgan Kaufmann, San Francisco, 1993

[Muk81] Mukaidono M. : A set of independent and complete axioms for a fuzzy algebra (Kleene Algebra). *Proc. IEEE 11th Int. Symp. on Multiple-Valued Logic* (Hrg.: S. C. Lee), Oklahoma, S. 27-34, 1981

[MUV84] Maier D., Ullman J.D. und Vardi M.Y.: On the foundations of the universal relation model. *ACM Trans. on Database Systems* 9 (2), S. 283-308, 1984

[NG78] Nicolas J.M. und Gallaire H.: Data bases: theory vs. interpretation. In: *Logic and Databases* (Hrg.: H. Gallaire und J. Minker), Plenum Press, S. 33-54, 1978

[NPS91] Negri M., Pelagatti S. und Sbatella L.: Formal semantics of SQL queries. *ACM Trans. on Database Systems* 16 (3), S. 513-534, 1991

[OO93] Ola A. und Ozsoyoglu G.: Incomplete relational database models based on intervals. *IEEE Trans. on Knowledge and Data Engineering* 5 (2), S. 294-308, 1993

[Rei78a] Reiter R.: On closed world databases. In *Logic and Data Bases* (Hrg.: H. Gallaire und J. Minker), Plenum Press, S. 55-76, 1978

[Rei78b] Reiter R.: Deductive question-answering on relational data bases. In *Logic and Data Bases* (Hrg.: H. Gallaire und J. Minker), Plenum Press, S. 149-177, 1978

[Rei80] Reiter R.: Equality and domain closure in first-order databases. *J. of the ACM* 27 (2), S. 235-249, 1980

[Rei84] Reiter R.: Towards a logical reconstruction of relational database theory. In: *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages* (Hrg.: M.L. Brodie et al.), Springer Verlag, Berlin, S. 191-233, 1984

[Rei86] Reiter R.: A sound and sometimes complete query evaluation algorithm for relational databases with null values. *J. of the ACM* 33 (2), S. 349-370, 1986

[Res69] Rescher N.: *Many-Valued Logic.* McGraw-Hill, New York, 1969

[Ris77] Rissanen J.: Independent components of relations. *ACM Trans. on Database Systems* 2 (4), S. 317-325, 1977

[RKS89] Roth M.A., Korth H.F. und Silberschatz A.: Null values in nested relational databases. *Acta Informatica* 26, S. 615-642, 1989

[RKS91] Roth M.A., Korth H.F. und Silberschatz A.: Addendum to null values in nested relational databases. *Acta Informatica* 28, S. 607-610, 1991

[Rob65] Robinson J.A.: A machine oriented logic based on the resolution principle. *J. of the ACM* 12 (1), S. 23-41, 1965

[Sch87] Schöning U.: *Logik für Informatiker.* B.I. Wissenschaftsverlag, Mannheim, 1987

[Sik81] Siklossy L.: Efficient query evaluation in relational databases with missing values. *Inform. Proc. Letters* 13 (4,5), S. 160-163, 1981

[Sin68] Sinowjew A.A.: *Über mehrwertige Logik. Ein Abriß.* Deutscher Verlag der Wissenschaften, Berlin, 1968

[Sin70] Sinowjew A.A.: *Komplexe Logik.* Vieweg, Braunschweig, 1970

[Smu68] Smullyan R.T.: *First Order Logic.* Springer-Verlag, Berlin, 1968

[TG92] Tansel A.U. und Garnett L.: On Roth, Korth, and Silberschatz's extended algebra and calculus for nested relational databases. *ACM Trans. on Database Systems* 17 (2), S. 374-383, 1992

[Tha89] Thalheim B.: On semantic issues connected with keys in relational databases permitting null values. *J. Inf. Process. Cybern. EIK* 25 (1/2), S. 11-20, 1989

[Tra50] Trachtenbrot B.A.: Über die algorithmische Unlösbarkeit des Entscheidungsproblems für endliche Individuenbereiche (russ.). *Dokl. Akad. Nauk SSSR* 70, S. 569-572, 1950; Übersetzung in: *American Mathem. Soc. Translation Series 2* 23, S. 1-5, 1963

[TS88] Topor R.W. und Sonenberg E.A.: On domain independent databases. In: *Foundations of Deductive Databases and Logic Programming* (Hrg.: J. Minker), Kapitel 6, S. 217-240, Morgan Kaufmann, San Francisco, 1988

[Ull89] Ullman J.D.: *Principles of Database and Knowledge-Base Systems.* Band 2, Computer Science Press, Rockville, 1989

[Var86] M.Y. Vardi: Querying logical databases. *J. Comput. System Sci.* 33 (2), S. 142-160, 1986

[Vas79] Vassiliou Y.: Null values in data base management: a denotational approach. *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, Boston, S. 162-169, 1979

[Vas80] Vassiliou Y.: Functional dependencies and incomplete information. *Proc. 6th VLDB Conf.*, Montreal, S. 260-269, 1980

[Win88] Winslett M.: A model-based approach to updating databases with incomplete information. *ACM Trans. on Databases Systems* 13 (2), S. 167-196, 1988

[WN88] Williams M.H. und Nicholson K.A.: An approach to handling incomplete information in databases. *The Computer Journal* 31 (2), S. 133-140, 1988

[YC88] L.Y. Yuan and D.-A. Chiang: A sound and complete query evaluation algorithm for relational databases with null values. *Proc. of the ACM SIGMOD Int. Conf. on Management of Data, Sigmod Record* 17 (3), Chicago, S. 74-81, 1988

[YH85] Yahya A. und Henschen L.J.: Deduction in non-Horn database. *J. Automated Reasoning* 1 (2), S. 141-160, 1985

[Yue91] Yue K.: A more general model for handling missing information in relational databases using a 3-valued logic. *ACM SIGMOD RECORD* 20 (3), S. 43-49, 1991

[Zan82] Zaniolo C.: Database relations with null values (extended abstract). *Proc. 1st ACM Symp. on Principles of Database Systems*, Los Angeles, S. 27-33, 1982

[Zan84] Zaniolo C.: Database relations with null values. *J. Comput. System Sci.* 28 (1), S. 142-166, 1984

Verzeichnis der wichtigsten Symbole und Abkürzungen

Attr(RT)	Attributmenge von RT	18
σ	DB-Schema	18
dom	Wertebereichsfunktion	18
z	DB-Zustand	18
$\mathfrak{Z}_\sigma$	Menge aller DB-Zustände zu σ	18
z_ω	ω-Zustand	18
$\geq$	Überdeckung	18,19,27
$\trianglerighteq$	Vervollständigung, Ausdehnung	18,19
Def(t)	Attributmenge, auf denen t total ist	19
$\leq$-inf(t,t')	Infimum bzgl. Überdeckungsrelation	19
t Δ t'	t verträglich mit t'	19,27
$\leq$-sup(t,t')	Suprenum bzgl. Überdeckungsrelation	19
$\sim$	überdeckungsäquivalent	19
$POSS_c$	Möglichkeitsfunktion Vervollständigung	22
$POSS_{ce}$	Möglichkeitsfunktion enge Ausdehnung	23
CWA	closed world assumption	24
OWA	open world assumption	24
ν	Belegungsfunktion	26
$POSS_V$	Möglichkeitsfunktion für V-Relationen	26
z_V	V-Zustand	26
$\geq_g$	genaue Überdeckung	27,54
Δ_s	stark verträglich	27
merge(t,t')	Verschmelzung	28
$\mathfrak{R}_\beta$	Menge aller Relationen über β	31
typ(x)	Attributmenge zu x	32
$\mathcal{GA}_{total}(q,z_p)$	gesicherte totale Antwort auf q in z_p	39
$\mathcal{GT}(q,z_\omega)$	Menge aller ges. Antworttupel zu q in z_ω	42
$\mathcal{GA}_{max}(q,z_\omega)$	gesicherte Maximum-Antwort auf q in z_ω	42
$\mathcal{GA}_{min/max}(q,z_\omega)$	gesicherte min/max-Antwort auf q in z_ω	43
$\mathcal{GA}_{s\text{-}uni}(q,z_\omega)$	strenge gesicherte uni-Antwort auf q in z_ω	45
$\mathcal{GA}_{uni}(q,z_\omega)$	gesicherte uni-Antwort auf q in z_ω	46
$\mathcal{GA}_{uni}(q,z_V)$	gesicherte uni-V-Antwort auf q in z_V	47
$\geq_i$	Informationsordnung	49
$\equiv_i$	informationsäquivalent	49
meet	Informationsschnitt	49
$\mathcal{GT}(q,z_V)$	Menge aller ges. Antworttupel auf q in z_V	50
$\mathcal{GA}_{max}(q,z_V)$	gesicherte Maximum-V-Antwort	51
$\mathcal{GA}_{min/max}(q,z_V)$	gesicherte min/max-V-Antwort	52
$\mathcal{GT}_\omega(q,z_V)$	Menge aller gesicherten (ω,V)-Antworttupel	53
$\mathfrak{A}_\omega(q,z_\omega)$	ω-Antwort	67
I_{sub}	spezielle Interpretation	73
$\omega_{\{\}}$	mengenwertiges Vorkommen von ω	74,92
$\mathfrak{A}_{sub}(q,z_\omega)$	sub-Antwort	78
$\cup_e, \setminus_e, []_e, \ldots$	erweiterte Operationen	85
$\mathfrak{A}_{sm}(e,z_\omega)$	sm-Antwort	86
$\cup_s, \setminus_s, []_s, \ldots$	s-Versionen der Algebraoperationen	89,113,120
$\cup_m, \setminus_m, []_m, \ldots$	m-Versionen der Algebraoperationen	89,113,120
$\mathfrak{A}_{switch}(e,z_V)$	Ergebnis der switch-Auswertung	122
$\mathfrak{W}$	Ergebnis einer Auswertung bei C-Relationen	133
$\mathfrak{A}_C(e,z_C)$	C-Antwort auf e in z_C	133
B_t	spezieller Boolescher Ausdruck zu t	134