Die Anwendung Generalisierter Quantoren in einem natürlichsprachigen Datenbank−Interface

Wolfgang Heinz
Johannes Matiasek

Österreichisches Forschungsinstitut für Artificial Intelligence
Schottengasse 3, A−1010 Wien
(Email: wolfgang%oefai@ai−vie.uucp, john%oefai@ai−vie.uucp)

Abstract

One main task of a natural language interface to a relational database is to map the semantic structures created by the parser onto database relations by translating them into statements of the query language. In this paper we present a method which employs an intermediate logical representation based on the theory of Generalized Quantifiers. These theoretical foundations provide the framework for an effective and straightforward implementation.

1. Vorbemerkungen

Ein natürlichsprachiges Datenbank−Interface läßt sich in verschiedene Komponenten gliedern. Zwei dieser Komponenten sind üblicherweise die Grammatik−Komponente für den natürlichsprachigen Teil und das Datenbank−Management−System mit seiner formalen Abfragesprache.

Die vorliegende Arbeit beschreibt Struktur und Funktionsweise einer Komponente, der Interpretationskomponente, die die Ergebnisse der grammatischen Analyse in Ausdrücke der formalen Datenbankabfragesprache transformiert. Als theoretische Basis für die Interpretationskomponente wählen wir die Theorie der Generalisierten Quantoren. Dies führt zu einer theoretisch begründeten Definition der Schnittstellen zu Grammatik und Datenbank. Primär linguistische Überlegungen, die auf eine semantische Restriktion der Generalisierten Quantoren zielen, führen auch zu einer Vereinfachung der Implementation.

Das hier beschriebene Verfahren wurde im Rahmen des deutschsprachigen Datenbank−Interface *DB−DIALOG* implementiert. Eine Gesamtdarstellung von *DB−DIALOG* findet sich in Trost, Buchberger, Heinz, Hörtnagl & Matiasek (1987).

2. Die Schnittstellen

2.1. Die Grammatik−Komponente

Die Aufgabe dieser Komponente ist die morphologische, syntaktische und semantische Analyse eines natürlichsprachigen Eingabesatzes, sowie die Weiterleitung des Ergebnisses dieser Analyse in einer geeigneten Darstellungsform an die Interpretationskomponente.

Typischerweise, wie auch im hier vorliegenden Datenbank−Interface, liegt das Ergebnis der Grammatikkomponente in einer baumartigen Frame−Darstellungsweise vor. Somit sollte eine Anfrage wie *(1)* eine Struktur wie *(2)* erzeugen:

(1) *Welche Patienten leiden an einer Infektionskrankheit?*

(2)
```
(Leiden−an   ((A1 (Patient    ... ))
              (A2 (Infektionskrankheit  ... ))
              (...)))
```

Zur genauen Identifikation (insbesondere bei Mehrfachauftreten eines Prädikates) erhält jedes Prädikat einen eindeutigen Identifier (*self*−Slot), zudem werden die Slots benannt. Eine Strukturierung der im *det*−Slot enthaltenen Information erfolgt weiter unten. Die solcherart erhaltene Darstellung

```
(3)   ((Prädikat      leiden−an)
       (self          x1    )
       (Argumente     ((a1    ((Prädikat  Patient)
                               (self       x2 )
                               (det        welch)
                               ....    ))
                      (a2      ((Prädikat  Infektionskrankheit)
                               (self       x3 )
                               (det        ein)
                               ....    ))))
       ....)
```

entspricht im wesentlichen den in DB−DIALOG verwendeten Caseframes. Eine genaue Beschreibung der Grammatik−Komponente und damit der obigen Strukturen findet sich in Trost, Buchberger & Heinz (1988).

Der Typ der Struktur aus *(3)* ist jedoch allgemein genug, um als Schnittstelle zu einem Gutteil der zur Zeit verwendeten Grammatikformalismen zu dienen. Beispielsweise erzeugen die meisten LFG−basierten Parser (cf. Bresnan (1982)) f−Strukturen, die mit *(2)* kompatibel sind, von HPSG (Creary & Pollard (1985)) werden vergleichbare Strukturen erzeugt, weiters findet sich die Prädikat−Argument−Struktur in den θ−Rollen der Government−Binding Theorie (cf. Chomsky (1981)).

2.2. Die Datenbank

Für unsere Betrachtungen beschränken wir uns auf relationale Datenbanken. Eine relationale Datenbank besteht aus *Tabellen* (die als *Prädikate* interpretiert werden können). Die *Zeilen* der Tabellen identifizieren die *Individuen*, auf die das Prädikat zutrifft. Die *Spalten* enthalten Ausprägungen von *Attributen.*

Zur Theorie des Relationenmodelles sei auf Codd (1970, 1979) verwiesen. Wichtig im Zusammenhang mit der beschriebenen Interpretationskomponente ist die Möglichkeit der Interpretation des Relationenkalküls in der Prädikatenlogik erster Ordnung (siehe Codd (1979:408f)).

Im Relationenkalkül sind, neben den mengentheoretischen Operationen, noch eine Reihe anderer Operationen definert (Restriktion, Projektion, Join), um aus den in der Datenbasis enthaltenen Tabellen neue Tabellen abzuleiten. Diese Operationen dienen dazu, Informationen aus der Datenbasis zu extrahieren, somit Fragen über den Inhalt der Datenbank zu beantworten. Als de−facto−Standard für relationale Datenbanken hat sich die Abfragesprache SQL herausgebildet (ISO (1986)). Das Ziel unserer Interpretationskomponente ist daher eine Darstellung der Abfrage in SQL.

Zur Illustration der Abfrage *(1)* das daraus generierte SQL−Statement

```
(4)   SELECT  DISTINCT X2.PNR, X2.NAME
      FROM    PERSON X2, IST_KRANK X1, KRANKHEIT X3
      WHERE   X3.INF = 'J' AND
              X3.KNR = X1.KNR AND
              X1.PNR = X2.PNR;
```

wobei dieser Abfrage das Datenbankmodell

```
(5)   Tabelle:  PERSON     Spalten:  PNR     Personen−Nummer  (Schlüssel)
```

		NAME	*Personen−Name*
Tabelle:	*KRANKHEIT Spalten:*	*KNR*	*Krankheits−Nummer (Schlüssel)*
		NAME	*Krankheits−Bezeichunung*
		INF	*infektiös (J/N)*
Tabelle:	*IST_KRANK Spalten:*	*PNR*	*Personen−Nummer (Schlüssel)*
		KNR	*Krankheits−Nummer (Schlüssel)*

zugrundeliegt. In *(4)* werden die charakteristischen Attribute von Personen ausgewählt, die in der *IST_KRANK* Tabelle einen Eintrag haben, dessen Krankheits−Nummer die Krankheitsnummer einer Infektionskrankheit (einer Krankheit mit *INF='J')* in der Tabelle *KRANKHEIT* ist.

3. Generalisierte Quantoren

Generalisierte Quantoren wurden im richtungsweisenden Artikel von Barwise und Cooper (1981) eingeführt. Sie basieren auf der Idee, daß *NP−Denotationen* die eigentlichen Quantoren sind und nicht die *Determinatoren* der NP, also 'alle Patienten' ist der Quantor, nicht 'alle'. Im Satz

(6) Alle Patienten leiden.

können wir die NP *Alle Patienten* mit dem Verb *leiden* kombinieren. Die zugehörige semantische Operation wäre, zu verifizieren, ob die Eigenschaft *LEIDEN* in der Menge der Eigenschaften, die alle Patienten besitzen, vorhanden ist. Die Denotation von *alle Patienten* muß also eine Menge von Eigenschaften sein. Nehmen wir an, daß Nomina wie *Patient* Eigenschaften als Denotation haben, so muß in einer kompositionellen Semantik die Denotation eines Determinators wie *alle* eine Funktion von Eigenschaften in Mengen von Eigenschaften sein. Formal kann man die Interpretation von *(6)* daher wie folgt darstellen:

(7) [*ALLE* [*PATIENTEN*]] [*LEIDEN*]

Da man Eigenschaften extensional als Mengen von Individuen auffassen kann und Mengen durch ihre charakteristische Funktion als *Funktion*, ist *(7)* wie folgt zu lesen:

<table>
<tr><td>ALLE</td><td>PATIENTEN</td><td>LEIDEN</td></tr>
<tr><td>< P, < P, t >></td><td>P</td><td rowspan="2">P</td></tr>
<tr><td colspan="2">< P, t ></td></tr>
<tr><td colspan="3">t</td></tr>
</table>

(8.1) *PATIENTEN* *ist eine Eigenschaft* *(Typ* **P***, als Abkürzung für* **<e,t>***)*

(8.2) *LEIDEN* *ist eine Eigenschaft* *(Typ* **P***)*

(8.3) *ALLE* *ist eine Funktion von Eigenschaften in Mengen von Eigenschaften* *(Typ* **<P,<P,t>>***)*

(8.4) *wird nun ALLE auf PATIENTEN angewandt, so ergibt sich als Denotation von ALLE PATIENTEN eine Menge von Eigenschaften, nämlich die Menge der Eigenschaften, die alle Patienten besitzen* *(Typ* **<P,t>***)*

(8.5) *(ALLE PATIENTEN), der Generalisierte Quantor, wird auf die Eigenschaft LEIDEN angewandt und ergibt somit einen Wahrheitswert.*

Wir verwenden dabei die Schreibweise einer Typenlogik, wie sie aus Montague (1973) bekannt ist. Hiebei steht **e** für Individuum (entity), **t** für Wahrheitswert (truth−value) und $<\alpha, \beta>$ für eine Funktion mit Argument vom Typ α und Wert vom Typ β.

Eine Formel über der Sprache der Generalisierten Quantoren hat also die Grundstruktur

(9) [*Det* [*B*]] [*Z*]

d.h. der Generalisierte Quantor aus dem Determinator *Det* und dem Basisprädikat *B* wird auf das Zentralprädikat *Z* angewandt. Die Gültigkeit einer solchen Formel wird aus dem Vergleich von *B* und *Z* ermittelt, wobei sich die Ermittlungsvorschrift aus *Det* ergibt.

Werden mehrstellige Prädikate verwendet, so werden die Argumente explizit mit dem λ−Operator angesprochen. Als Beispiel etwa

(10) Mehr als 2 Patienten leiden an einer Infektionskrankheit.

mit den beiden Interpretationen

(11) a) [*MEHR ALS 2* [**Patienten**]] λ*x*[[*EINE* [**Infektionskrankheit**]] λy[**leiden−an** *(x,y)*]]
b) [*EINE* [**Infektionskrankheit**]] λy[[*MEHR ALS 2* [**Patienten**]] λ*x*[**leiden−an** *(x,y)*]]

wobei die beiden Lesarten die beiden möglichen Scopus−Interpretationen ergeben.

Neben der Verallgemeinerung der NP−Denotation − NP als Typ **< P, t >**, zusammengesetzt aus dem Determinator mit dem Typ **< P,< P, t > >** und dem Nomen mit dem Typ **P** − ist ein weiteres wichtiges Ergebnis von Barwise & Cooper (1981) die semantische Charakterisierung von natürlichsprachigen Determinatoren. Keenan & Stavi (1986) führen diese Arbeit weiter und versuchen, die mögliche Denotation von Determinatoren, nämlich alle Funktionen von Eigenschaften in Mengen von Eigenschaften (**P** → **P***), einzuschränken. Dabei kommen sie zu folgender Charakterisierung der natürlichsprachigen Determinatoren:

(12) (a) Es gibt eine Menge von Basisdeterminatoren (Basic$_{Det}$) *bestehend aus:*
1) **EVERY**
2) KARD (Basic Cardinality Det): **AT−LEAST** *n,* $n \in$ **N**
3) POSS (Basic Possessive Det): ***J*'s** *n* **OR MORE,** $n \in$ **N**, *J Individuum*
(b) D$_{Det}$ *ist die kleinste Menge, die* Basic$_{Det}$ *enthält und bezüglich Vereinigung, Schnitt und Komplement abgeschlossen ist.*

Die formalen Definitionen zu *(12)* entnehme man Keenan & Stavi (1986:261−264). Eine alternative Charakterisierung von D$_{Det}$ ergibt sich aus dem Konservativitätstheorem, cf. Keenan & Stavi (1986:276).

(13) **Konservativitätstheorem:**
D$_{Det}$ ist genau die Menge der konservativen Funktionen von **P** → **P*.**
Eine Funktion **f** *ist konservativ g.d.w.* **p, q** *Eigenschaften:* $\mathbf{p} \in \mathbf{f(q)} \Leftrightarrow (\mathbf{p} \wedge \mathbf{q}) \in \mathbf{f(q)}$

Ein Beispiel für die Gültigkeit des Konservativitätstheorems (mit **f**=*ALLE,* **q**=*PATIENT,* **p**=*LEIDEN)*

(14) a) Alle Patienten leiden g.d.w. gilt: alle Patienten sind Patienten und leiden.
b) [*ALLE* [*PATIENT*]] [*LEIDEN*] ⇔ [*ALLE* [*PATIENT*]] [*PATIENT* ∧ *LEIDEN*]

Für die Anwendung der Generalisierten Quantoren in natürlichsprachigen Datenbank−Interfaces ist das Konservativitätsuniversale *(15)* von Bedeutung (Keenan & Stavi (1986:260)).

(15) Extensionale Determinatoren in allen Sprachen werden immer durch konservative Funktionen interpretiert.

Aus dem Konservativitätstheorem ergibt sich, daß die Ermittlung der Gültigkeit von *(9)* nicht mehr von B und Z, sondern nur mehr von B und $B \cap Z$ abhängt. Diese Mengen lassen sich, wie in Hamm (1986:217ff.) zusammengefaßt, weiter einschränken, nämlich auf $B \setminus Z$ und $B \cap Z$. Letztlich ergibt sich sogar

(16) *Bei endlichen Modellen ist zur Ermittlung der Gültigkeit der Formel nur die Betrachtung der Kardinalitäten der Mengen* $B \setminus Z$ *und* $B \cap Z$ *erforderlich.*

4. Die Interpretationskomponente

Die vorangegangenen Kapitel bilden den Rahmen für unseren Vorschlag zur Struktur einer Interpretationskomponente. Unsere Interpretationskomponente umfaßt zwei Teile, die durch eine Zwischenstruktur, die **Logische Form**, verbunden sind. Diese Logische Form hat die Struktur einer Formel über der Sprache der Generalisierten Quantoren.

Für den ersten Schritt, die Erstellung der Logischen Form aus dem Caseframe, spezifizieren wir die Übersetzung der Domänenprädikate in Datenbank−Prädikate und Anforderungen an den Informationsgehalt des Determiner−Slots im Caseframe.

Im zweiten Schritt wird die Logische Form in ein SQL−Statement transformiert und die Übersetzung der Determinatoren durch SQL−Konstrukte dargestellt. Diesen Schritt zu isolieren ist auch eine Motivation für die Einführung der Logischen Form, da einerseits trotz der Bestrebungen, SQL zu normieren, noch eine Reihe von SQL−Dialekten existieren, andererseits SQL auch eine Reihe von Eigenheiten aufweist, die sich nicht zwingend aus dem Relationenkalkül ergeben (cf. Date (1987)). Es erweist sich daher als günstig, Eigenheiten der Abfragesprache in einem eigenen Modul zu verpacken.

4.1. Von der Grammatik zur Logischen Form

4.1.1. Die Transformation der Prädikate

Die Prädikate der Caseframes entsprechen Prädikaten der logischen Domänenmodellierung und müssen daher nicht notwendigerweise mit den in Relationen manifestierten Prädikaten der Datenbankmodellierung übereinstimmen. Es ist daher notwendig, eine eindeutige Zuordnungsvorschrift zwischen den beiden Prädikatsarten zu geben. Dies wollen wir in diesem Kapitel anhand einer Beispieltransformation des Caseframes *(3)* in eine Logische Form auf Basis des Datenbankmodells *(5)* zeigen.

Der Caseframe *(3)* ist folgendermaßen zu interpretieren: Das Basisprädikat **leiden−an** hat als Argumente die Konstrukte *a1* und *a2* mit den zugrundeliegenden Prädikaten **Patient** bzw. **Infektionskrankheit**. **x1, x2** und **x3** sind die jeweiligen Instanzen der Prädikate. In Kurzform ist *(3)* als **x1 (x2, x3)** zu schreiben. Die Bindungen sind somit implizit durch die *self*−Namen gegeben.

Transformieren wir nun *(3)* in eine Darstellung, die als Prädikate bereits Datenbank−Prädikate enthält, so sind zwei Dinge zu tun − einerseits die Übersetzung der logischen Prädikatsnamen in Datenbank−Prädikatsnamen, d.h. Relationsnamen, und andererseits die Explizitmachung der Bindungen in der Datenbank durch Join−Operationen zwischen den Relationen.

Nehmen wir als einfachste Form der Übersetzung der Prädikate das Beispiel **Patient**. Diese Eigenschaft wird in der Datenbank direkt durch die Zugehörigkeit eines Tupels zur Tabelle *PERSON* festgelegt. Ein Tupel und damit ein Individuum im Sinne unseres Datenmodells ist durch seinen Schlüssel eindeutig identifiziert. **Patient (x2)**, d.h. x2 ist Patient, wird in der Datenbank durch 'Das Tupel x2 der Relation *PERSON*, identifiziert durch seine *PNR*' dargestellt.

Das Basisprädikat von *(3)* ist die Beziehung **leiden−an**, ein mehrstelliges Prädikat. Es ist in unserer Datenbank als Assoziationsrelation *IST_KRANK* definiert, d.h. es stellt eine Beziehung zwischen den Kernrelationen *PERSON* und *KRANKHEIT* mittels der Fremdschlüssel *PNR* bzw. *KNR* dar. Die

Argumentstruktur der Caseframes wird in der Datenbank nun mittels dieser Fremdschlüsselbeziehungen dargestellt, in unserem Beispiel also *X1.PNR = X2.PNR* und *X1.KNR = X3.KNR.* Zu den Begriffen Kernrelation, Assoziationsrelation und Fremdschlüssel cf. Codd (1979).

Der erste Schritt unserer Transformation des Caseframe ist also die Übersetzung der logischen Prädikate in Relationsnamen und die Darstellung der Bindung zwischen Prädikaten und Argumenten mithilfe von Join−Bedingungen auf der Datenbank−Ebene.

Die Übersetzung von logischen in Datenbank−Prädikate ist dabei nicht in jedem Fall so trivial wie in den oben angeführten Beispielen. Ein logisches Prädikat kann etwa in mehrere verschiedene Datenbank−Prädikate transformiert werden, was eine Ambiguität auf der Datenbank−Ebene erzeugt, die auf der logischen Ebene noch nicht vorhanden ist. Als Beispiel sei die Modellierung des Begriffs 'Person' als logisches Prädikat **Person** angeführt, das in der Datenbank die Entsprechungen in Tabellen wie *PATIENT* und *ARZT* haben kann. Dies führt dazu, daß die Transformation nicht als Isomorphismus aufgefaßt werden kann.

Weitere häufige nichttriviale Übersetzungen sind Abbildungen in abgeleitete Relationen, z.B. Relationen, die durch Restriktion aus Basisrelationen gewonnen werden können. Als Bespiel dafür tritt in *(3)* die Übersetzung von Infektionskrankheit in die Relation *KRANKHEIT* mit der Restriktion auf INF = 'J' auf, d.h. die Beschränkung der Relation auf die Tupel und damit Individuen, die die Eigenschaft *infektiös* besitzen.

4.1.2. Die Determinatoren

Das Beispiel *(1)* ist in seiner Struktur nun nicht so einfach, daß es in der Form *P(x,y)* dargestellt werden könnte. Es sollte vielmehr nach den Erkenntnissen aus Kapitel 3. eine Form wie

(17) [*?WELCHE* [**Patienten**]] λ*x*[[*EINE* [**Infektionskrankheit**]] λy[**leiden−an** *(x,y)*]]

erhalten, wobei allerdings anstelle der logischen Prädikate bereits die Prädikate der Datenbank (Relationen) eingesetzt sein sollten. Die Grobstruktur von *(17)* ist bereits in *(3)* gegeben − das Zentralprädikat von *(17)* entspricht der Wurzel des Caseframe−Baumes, die Generalisierten Quantoren haben ihre Entsprechung in den Tochterknoten ('Argumenten'). In *(3)* fehlt jedoch als wesentliche Information die über die Determinatoren.

Die Ergebnisse aus Kapitel 3. erlauben nun, die Darstellungsform der Determinatoren sehr einfach zu halten, da diese auf wenige Primitiva zurückgeführt werden können. Aus der Grammatikkomponente erwarten wir also eine Det−Information, die nach *(12)* aus Angaben über die *Kardinalität* oder *ALLE* besteht. Die Behandlung der Basispossessiva erübrigt sich (in DB−DIALOG aus unabhängigen Gründen), da Possessivkonstruktionen in äquivalente Darstellungen transformiert werden können, für die im Rahmen der Grammatik ohnedies eine Interpretation vorliegen muß. Als Beispiel dazu etwa

(18) *a) Müllers Stethoskop* → *das Stethoskop von Müller*
(= das Stethoskop, das Müller gehört)
b) meine 3 Patienten → *die 3 Patienten von mir*
(= die 3 Patienten, die ich behandle)

Da die Possessor−Relation, die auch in der Definition von *(12a3)* vorkommt, auch vom Nomen abhängig ist (wie *(18)* zeigt), und außerdem von der Grammatik−Komponente ohnehin für die Postmodifikator−Position eines Genitivs oder einer PP benötigt wird, reduziert diese Transformation die Menge der Basis−Determinatoren auf *EVERY* und *KARD*. Wir verlangen also von der Grammatikkomponente eine Determinator−Information, die sich aus Booleschen Kombinationen dieser Determinatoren zusammensetzt. Dazu einige Beispiele (in der Notation von *(12)*):

(19) *a) alle* → *EVERY*
b) zwei → *AT−LEAST 2*
c) weniger als 3 → *(AT−LEAST 3)'*

d) zwischen 2 und 4 $\rightarrow$ *(AT−LEAST 2)* $\wedge$ *(AT−LEAST 5)'*
e) ein $\rightarrow$ *AT−LEAST 1*
f) kein $\rightarrow$ *(AT−LEAST 1)'*

Laut *(16)* lassen sich die Determinatoren auf Kardinalitätsüberprüfungen von Mengen zurückführen: *(19a)* auf $|B \setminus Z| = 0$, *(19b−f)* auf Boolesche Kombinationen von $|B \cap Z| \geq n$. Die Mengen, deren Kardinalitäten festzustellen sind, sind also einerseits die Menge der Individuen, auf die sowohl das Zentralprädikat Z als auch das Basisprädikat B des Generalisierten Quantors zutrifft (die *Schnittmenge* $B \cap Z$ der beiden Prädikate), und andererseits die Menge der Individuen, auf die zwar das Basisprädikat des Generalisierten Quantors, nicht aber das Zentralprädikat zutrifft (die *Differenzmenge* $B \setminus Z$ der beiden Prädikate).

Wir wollen dies an zwei Beispielen erläutern. In

(20) Alle Patienten leiden.

muß für die Interpretation des Determinators *ALLE* und damit des Generalisierten Quantors *ALLE PATIENTEN* nur die Kardinalität der Menge überprüft werden, die Individuen enthält, die Patienten sind, aber **nicht** leiden. Diese Menge muß Kardinalität 0 haben, es muß also gelten:
$|B \setminus Z| = 0$. In

(21) Mindestens drei Patienten leiden.

muß für die Interpretation des Determinators *MINDESTENS DREI* und damit des Generalisierten Quantors *MINDESTENS DREI PATIENTEN* nur die Kardinalität der Menge überprüft werden, die Individuen enthält, die **sowohl** Patienten sind **als auch** leiden. Diese Menge muß eine Kardinalität von zumindest 3 haben, es muß also gelten: $|B \cap Z| \geq 3$.

Wir verwenden in DB−DIALOG nun folgende Notation: Der Determinator in der Caseframe−Form besteht aus mehreren Subslots mit den Namen *fct, rel, const* und *interrog*, die an anderer Stelle aus unabhängigen Gründen ebenfalls gebraucht werden.

Der Slot *fct* enthält im unbefragten Determinator−Fall entweder **all** oder **count**. **all** steht für *EVERY* bzw. $|B \setminus Z| = 0$ und gibt somit Überprüfungsvorschrift und zugrundeliegende Menge an. Die Slots *rel* und *const* sind in diesem Falle irrelevant. Ist der Basisdeterminator *AT−LEAST* und damit die Überprüfung der Form $|B \cap Z| \geq n$, so steht im *fct*−Slot **count**. Da wir aus Gründen der Vereinfachung nicht nur $\geq$ als Relationsoperator zulassen wollen, nimmt der Subslot *rel* den Relationsoperator auf. Im Subslot *const* wird die Vergleichskonstante *n* aufgeführt. Mit dieser Schreibweise lassen sich auch die meisten Booleschen Kombinationen auf einen Basisausdruck zurückführen. *(22)* zeigt die Übersetzung der Determinatoren aus *(19)* in DB−DIALOG.

(22)

a) alle	$\rightarrow$	*fct:* **all**		
b) zwei	$\rightarrow$	*fct:* **count**	*rel:* $\geq$	*const:* **2**
c) weniger als 3	$\rightarrow$	*fct:* **count**	*rel:* $<$	*const:* **3**
d) zwischen 2 und 4	$\rightarrow$	*fct:* **count**	*rel:* **between**	*const:* **(2, 4)**
e) ein	$\rightarrow$	*fct:* **count**	*rel:* $\geq$	*const:* **1**
f) kein	$\rightarrow$	*fct:* **count**	*rel:* $<$	*const:* **1**

Hinsichtlich der *Befragung* sei folgendes angemerkt: Ähnlich wie Hamblin (1973) betrachten wir die Denotation einer Frage als die Menge der Individuen des entsprechenden Typs, die − in die 'offene Variable' der Frage eingesetzt − die Frage zu einer wahren Aussage machen. Zur Interpretation von Fragen cf. Hamblin (1973), Karttunen (1977). Wir behandeln daher die Fragedeterminatoren ähnlich wie alle anderen und tragen in den Funktionsslot *fct* die Art der Befragung ein − nach der Anzahl **(count)** oder der Aufzählung **(value)** der Individuen, d.h. nach der Kardinalität der Menge oder der Menge selbst.

In der Form *(23)* entspricht *(3)* nun der Schnittstellendefinition zwischen Grammatikkomponente und Interpretationskomponente.

(23) *((Prädikat* **leiden−an***)*
(self **x1** *)*
(Argumente ((a1 ((Prädikat **Patient***)*
(self **x2** *)*
(Det ((fct **value***) (interrog* **T***))*
....))
(a2 ((Prädikat **Infektionskrankheit***)*
(self **x3** *)*
(Det ((fct **count***) (rel* $\geq$ *) (const* **1** *))*
....))))
....)

4.2. Die Struktur der Logischen Form

Aus der Übersetzung der logischen Prädikate in Datenbank−Relationen, der Darstellung der Determinatoren und der Linearisierung der Baumstruktur ergibt sich nun

(24) [[*VALUE, ?*] (*PERSON(x2)*)]
[[*COUNT,* $\geq$*, 1*] (*KRANKHEIT(x3)* $\wedge$ *x3.INF='J'*)]
[*IST−KRANK(x1)* $\wedge$ *x1.PNR=x2.PNR* $\wedge$ *x1.KNR=x3.KNR*]

als Logische Form zu *(3)* in Form der Generalisierten Quantoren. Die Determinatoren (gemäß der in 4.1.2. eingeführten *det*−Struktur, hier verkürzt notiert) bestimmen die Anwendung der Quantoren auf die Prädikate. Die Bindungen sind über die Join−Bedingungen als Restriktionen gegeben und führen, da ein Join zwischen zwei Relationen, die Eigenschaften eines Individuums charakterisieren, einem Durchschnitt über die Individuenmengen entspricht, zu einem Prädikat, das der Schnittmenge des Zentralprädikats mit dem Basisprädikat des Generalisierten Quantors entspricht.

Aus der Anwendbarkeit des Konservativitätstheorems folgt die Zulässigkeit dieser Darstellung: die Anwendung eines Generalisierten Quantors auf ein Prädikat liefert laut *(13)* dasselbe Ergebnis wie die Anwendung des Generalisierten Quantors auf das Prädikat *geschnitten mit dem Basisprädikat* des Generalisierten Quantors.

4.3. Von der Logischen Form zum SQL

Ein SQL−Query−Statement besteht im wesentlichen aus zwei Teilen, einer SELECT−Klausel und einer Tabellenbeschreibung (cf. ISO (1986:55)). Die Tabellenbeschreibung enthält dabei u.a. die FROM−Klausel, die die beteiligten Relationen angibt, und die optionale WHERE−Klausel, die die Restriktionen beschreibt. In der WHERE−Klausel können Subselects auftreten, Bedingungen, die mithilfe eines weiteren eingebetteten Query−Statements formuliert werden können.

Betrachten wir die Struktur der oben definierten Logischen Form, so fällt die Parallelität zur Struktur eines Query−Statements auf: dem Prädikat des Generalisierten Quantors entspricht eine Tabellenbeschreibung und dem Determinator die SELECT−Klausel. Die Übersetzung einer Struktur wie *(24)* in ein korrespondierendes SQL−Statement ergibt sich daher ganz natürlich:

(25) *a) Übersetze den Determinator in die entsprechende SELECT−Klausel.*
b) Übersetze das Basisprädikat des Generalisierten Quantors in die Tabellenbeschreibung, wobei die Relationsnamen in die FROM−Klausel eingehen und die Restriktionen in die WHERE−Klausel aufgenommen werden.
c) Erweitere die WHERE−Klausel um eine Restriktion, die sich aus der Übersetzung des nächsten Generalisierten Quantors, falls ein solcher vorhanden ist, ergibt. Ist das Zentralprädikat erreicht, folgt diese Restriktion aus der Übersetzung des Satzoperators in eine SELECT−Klausel und des Satzprädikats in eine Tabellenbeschreibung.

Somit ergibt sich aus *(24)* das SQL−Statement

(26)

```
SELECT  X2.PNR, X2.NAME
FROM    PERSON X2
WHERE   EXISTS   (SELECT *
                  FROM    KRANKHEIT X3
                  WHERE   X3.INF = 'J' AND
                          EXISTS   (SELECT *
                                    FROM IST_KRANK X1,
                                    WHERE  X3.KNR = X1.KNR AND
                                           X1.PNR = X2.PNR ));
```

In *(26)* ist der Ausdruck *'EXISTS (SELECT *...)'* als *'1 ≤ (SELECT COUNT (*) ...)'* zu verstehen, was genau die im Determinator angeführte Bedingung wiedergibt. Da unter gewissen Voraussetzungen Join−Bedingungen alleine äquivalente Ergebnisse wie EXISTS−Subselects liefern, würde DB−DIALOG das SQL−Statement *(26)* nur virtuell erzeugen, als Ergebnis jedoch die 'optimierte' Form *(4)* liefern.

Um die Zulässigkeit der obigen Übersetzung zu zeigen, verwenden wir wiederum Ergebnisse aus Kapitel 3. In unseren SQL−Statements wollen wir die Anwendung von Generalisierten Quantoren auf Prädikate wiedergegeben sehen. Wir müssen daher in SQL formulieren, wie wir eine Menge als Element einer Menge von Mengen eruieren können. Dies wäre ein etwas komplexes Unterfangen. Da wir jedoch auf der Basis von implementierten Datenbanken arbeiten und diese per definitionem endlich sind, ist auch das Modell, in dem unsere Interpretation stattfindet, endlich. Diese Voraussetzung erlaubt, *(16)* anzuwenden, nämlich die Anwendungsbedingungen auf Prüfungen von Kardinalitäten von Mengen zurückzuführen, was in SQL recht einfach ist.

Wir übersetzen die in *(19)* eingeführten Determinatoren ziemlich direkt in Konstrukte von SQL:

(27)

alle	*→ 0 = (SELECT COUNT(*) ... WHERE ... NOT ...)* bzw.
	*→ NOT EXISTS (SELECT * ... WHERE ... NOT ...)*
zwei	*→ 2 ≤ (SELECT COUNT (*) ...)*
weniger als 3	*→ 3 > (SELECT COUNT (*) ...)*
kein	*→ 0 = (SELECT COUNT (*) ...)* bzw.
	*→ NOT EXISTS (SELECT * ...)*

Die Übersetzungen in *(27)* sind, bis auf die Übersetzung von *alle*, trivial, da sie die Kardinalität der *Schnittmengen* überprüfen, die durch die Join−Bedingungen (vgl. deren Motivation aus Kapitel 4.2.) bereits gegeben sind. *Alle* hingegen ist ein Determinator, bei dem die Kardinalität der *Differenzmenge* ($B \setminus Z$) relevant ist. Aus der Definition

(28) $B \setminus Z \equiv B \cap Z'$

folgt nun, daß der Schnitt mit dem *Komplement* von *Z* gebildet werden muß. Der Schnitt ist wieder mittels der Join−Bedingungen zu formulieren, die *Z* entsprechenden Bedingungen der WHERE−Klausel müssen jedoch *negiert* werden. Diese Überlegungen führen auf dasselbe Ergebnis wie Überlegungen, die sich aus dem Fehlen eines Allquantors in SQL ergeben. Dieser ist durch die Äquivalenz

(29) $\forall x\, P(x) \Leftrightarrow \neg\exists x\, \neg P(x)$

gegeben, d.h. beim Transformationsalgorithmus muß sowohl der aktuelle Determinator übersetzt als auch der nächsttiefere Determinator (oder der Satzoperator, der entweder leer oder z.B. eine Satznegation sein kann) negiert werden.

5. Zusammenfassung

Wir haben eine Methode vorgelegt, die die Transformation einer Darstellung des semantischen Gehalts einer Datenbankanfrage in Caseframe−Notation in einen Ausdruck der Datenbankabfragesprache SQL durchführt. Dazu bedienen wir uns einer Zwischenstruktur, der Logischen Form, in der

Notationsweise der Generalisierten Quantoren. Die Rückführung der definierten Schnittstellen sowohl zur Grammatik−Komponente als auch zur Datenbank auf Generalisierte Quantoren−basierte Strukturen ermöglicht die Anwendung von zunächst linguistisch motivierten Generalisierungen und Einschränkungen auf das Aussehen und die Mächtigkeit der Strukturen. Die Anwendung der Beschränkungen und Definitionen, wie etwa des Konservativitätstheorems oder der Restriktion auf Kardinalitätsvergleiche, führen zu einfachen und theoretisch fundierten Übersetzungsschritten, die sowohl das Design dieser Interpretationskomponente klar gestalten als auch die Implementierung vereinfachen.

Anmerkung

Die Arbeit an DB−DIALOG wurde zusammen mit der Firma Software Management Gmbh, Wien durchgeführt und im Rahmen des Schwerpunktes S7 (Artificial Intelligence) des Förderungsprogramms 'Mikroelektronik und Informationsverarbeitung' der Österreichischen Bundesregierung gefördert.

Literatur

Barwise, J. & R. Cooper (1981) − 'Generalized Quantifiers and Natural Language'. In: *Linguistics & Philosophy 8.4: 159−219*

Bresnan, J. (ed.) (1982) − *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, Mass.

Chomsky, N. (1981) − *Lectures on Governement and Binding.* Foris, Dordrecht

Codd, E.F. (1970) − 'A Relational Model of Data for Large Shared Data Banks'. In: *Comm. of the ACM 13: 377−387*

Codd, E.F. (1979) − 'Extending the Database Relational Model to Capture More Meaning'. In: *ACM Transactions on Database Systems 4.4: 397−434*

Creary, L. & C. Pollard (1985) − 'A Computational Semantics for Natural Language'. In: *Proc. of 23rd Meeting of the ACL: 172−179*

Date, C.J. (1987) − 'Where SQL Falls Short': In. *Datamation May 1, 1987: 82−86*

Hamblin C.L. (1973) − 'Questions in Montague English'. In: *Foundations of Language 10: 41−53*

Hamm, F. (1986) − 'Generalisierte Quantoren und semantische Prinzipiensysteme'. In: *Linguistische Berichte 103: 201−223*

ISO (1986) − *Information processing systems − Database language SQL.* International Standard ISO/DIS 9075. International Organization for Standardization

Karttunen, L. (1977) − 'Syntax and Semantics of Questions'. In: *Linguistics & Philosophy 1,1: 3−44*

Keenan, E. & J. Stavi (1986) − 'A Semantic Characterization of Natural Language Determiners'. In: *Linguistics & Philosophy 9.3: 253−326*

Montague, R. (1973) − 'The Proper Treatment of Quantification in Ordinary English'. In: Hintikka, Moravcik & Suppes (eds.): *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics.* Reidel, Dordrecht

Trost, H., E. Buchberger, W. Heinz (1988) − 'On the Interaction of Syntax and Semantics in a Syntactically Guided Case-frame Parser'. In: *Proc. of 12th COLING: 677−682*

Trost, H., E. Buchberger, W. Heinz, Ch. Hörtnagl, J. Matiasek (1987) − 'Datenbank−DIALOG − A German Language Interface for Relational Databases'. In: *Applied Artificial Intelligence 1,2: 181−203*