# Ein Beweisverfahren für Datenbankprädikate

Stefan Böttcher

Fachbereich Informatik
Johann Wolfgang Goethe-Universität
Dantestraße 9, D-6000 Frankfurt/Main 1

**Kurzfassung**

Gezeigt wird, wie mit Methoden des automatischen Beweisens eine zentrale Aufgabe in Datenbanksystemen, die Synchronisation von Transaktionen, gelöst werden kann. Das Beweisverfahren basiert nicht auf Resolution [Robinson65], sondern erweitert eine Variante [Rosenkrantz80] der Sup-Inf-Methode. Die "Beweissuche" bzw. die "Konstruktion von Gegenbeispielen" wird auf die Suche nach Zyklen in verschiedenen Graphen reduziert. Dieses spezielle Beweisverfahren kann alle im vollen relationalen Kalkül [Codd72] formulierbaren Datenmengen testen.

## 1 Einleitung

Relationale Datenbanken gewinnen als Systemkomponenten in wissensbasierten Systemen zunehmend an Bedeutung. Die Leistungsfähigkeit solcher Systeme hängt wesentlich von der Unterstützung durch das zugrundeliegende Datenbanksystem ab [Appelrath85]. Dazu gehört u.a. die Kontrolle der Datenunabhängigkeit parallel ablaufender Aktionen auf der Datenbank (Transaktionen). Zweck dieser Kontrolle bzw. des Testverfahrens ist, zu garantieren, daß nur solche Transaktionen parallel ablaufen, die auf disjunkten Datenmengen arbeiten. ( Weitere, die Leistungsfähigkeit eines Datenbanksystems entscheidend bestimmende Komponenten, z.B. die Anfrageoptimierung, erfordern genau dieselben Tests auf Datenmengen [Munz79], [Böttcher85]. )

### 1.1 Wie können Datenbanksysteme Beweisverfahren einsetzen ?

Zwei Transaktionen dürfen parallel ablaufen, wenn sie auf disjunkten Datenmengen arbeiten [Eswaran76]. Dazu folgt ein Beispiel, notiert im relationalen Tupelkalkül von DBPL ([Mall84],[Edelmann84]), der Nachfolgesprache von PASCAL/R [Schmidt77]:
Eine Transaktion arbeitet auf "Tupeln t der Datenbankrelation (Relation) R, die im Attribut a1 einen kleineren Wert als 100 haben":

```
{ EACH t IN R : t.a1 < 100 }
```

Außerdem beansprucht eine andere Transaktion nur ein Exklusivzugriffsrecht auf "alle Tupel t der Relation R, die im Attribut a1 den Wert 500 einnehmen":

```
{ EACH t IN R : t.a1 = 500 }
```

Dann dürfen beide Transaktionen parallel ablaufen, denn die gemeinsam benutzte Datenmenge

```
{ EACH t IN R : ( t.a1 < 100 ) AND ( t.a1 = 500 ) }
```

ist leer.

Eine gemeinsam benutzte Datenmenge { EACH t IN R : P(t) } ist genau dann in jedem Datenbankzustand leer, wenn das Prädikat SOME t IN R ( P(t) ) unerfüllbar ist. Ein Prädikat heißt unerfüllbar, wenn es für alle Datenbankzustände den Wert FALSE annimmt.
( Im Unterschied zu Definitionen in [Klug83] und [Reiter84] ist der hier verwandte Erfüllbarkeitsbegriff unabhängig von den Integritätsbedingungen der Datenbank. Will man diese mit abprüfen, so muß man sie mit dem Prädikat konjunktiv verknüpfen. )

Betrachten wir N bereits aktive Transaktionen Ti, die auf Relationen Rk die Datenmengen

Datenmenge(Ti,Rk) = { EACH t IN Rk : PTi(t) }

benutzen: Eine weitere Transaktion Tx darf parallel laufen, wenn die Transaktionen auf jeder Relation paarweise disjunkte Datenmengen benutzen, wenn also für jede Relation Rk und für jede aktive Transaktion Ti gilt:

Datenmenge(Tx,Rk) n Datenmenge(Ti,Rk) = { } , also

{ EACH t IN Rk : PTx(t) AND PTi(t) } = { } .

Dafür muß jeweils von einem Theorembeweiser bewiesen werden : Das Prädikat

SOME t IN Rk ( PTx(t) AND PTi(t) ) ist unerfüllbar .

Jeder Synchronisationstest läßt sich auf einen Erfüllbarkeitstest in einem mehrsortigen Prädikatenkalkül zurückführen [Böttcher85]. Die Datenbankrelationen bilden paarweise disjunkte Sorten, da wertgleiche Tupel in verschiedenen Relationen z.B. in Update-Operationen einzeln änderbar sind, also als verschieden angesehen werden müssen.

## 1.2 Probleme bei der Verwendung bekannter Beweisverfahren

Entwicklungsziel bekannter Testverfahren ( [Andrews81], [Bibel81], [Bläsius81], [Hsiang83], [Kowalski75] ) ist der universell einsetzbare Beweiser für beliebige Prädikate 1.Ordnung, nicht jedoch ein spezieller Tester für Datenbankprädikate. Grundsätzlich stellen universelle Beweisverfahren das Terminationsproblem: Wegen der Unentscheidbarkeit des Prädikatenkalküls [Kleene71] gibt es für jeden vollständigen Entscheidungsalgorithmus Prädikate, für die er nicht terminiert. Man kann also **nicht** berechnen: Der Entscheidungsalgorithmus wird nach spätestens n Schritten terminieren. Aber ein Synchronisationstest muß in abschätzbarer Zeit terminieren. Deshalb wird in den folgenden Abschnitten ein spezielles Testverfahren entwickelt. Es verbindet Ideen der Sup-Inf-Methode ([Bledsoe77],[Shostak79]) mit einer Alternative zur Skolemisierung [Bibel82].

# 2 Matrixtests

Der Matrixtest testet die Erfüllbarkeit einer Matrix in disjunktiver Normalform; alle Tupelvariablen der Matrix müssen SOME-quantifiziert (kurz: SOME-Variablen) sein. Das Testprinzip ist:

> Eine Matrix in disjunktiver Normalform ist genau dann erfüllbar, wenn mindestens eine ihrer Konjunktionen erfüllbar ist.

Deshalb werden im folgenden Erfüllbarkeitstests für Konjunktionen entwickelt. Diese sind zudem der später beschriebenen Erweiterung des Matrixtests ähnlich.

## 2.1 Verzicht auf die Entscheidbarkeit, Zweifelsfälle

Läßt man die Vergleichsoperatoren ≠, = , >=, <=, > und < zu, ist das Problem, die Erfüllbarkeit einer Konjunktion festzustellen, NP-hard [Hunt79]. Ein Verfahren, das nicht "schnell" entscheidet, ist aber für viele Anwendungen nicht zu gebrauchen. Garey und Johnson [Garey79] empfehlen bei Problemen dieser Komplexität, eine Näherungslösung zu entwerfen, die in Polynomzeit in einer Klasse von häufigen Anwendungsfällen richtig entscheidet. Rosenkrantz und Hunt [Rosenkrantz80] schlagen eine Einschränkung der Sup-Inf-Methode vor, um in Polynomzeit O(Vergleichsanzahl hoch 3) die Erfüllbarkeit einer Konjunktion ohne ≠-Vergleiche zu testen. Dieses Verfahren läßt jedoch offen, wie man mit ≠-Vergleichen umgehen soll.

Allgemeiner gefragt: Wie soll man mit Prädikaten umgehen, für die ein gegebener Algorithmus nicht in akzeptabler Zeit entscheiden kann, ob sie erfüllbar sind oder nicht? Solche Prädikate werden im folgenden Zweifelsfälle genannt.

Die Synchronisation von Transaktionen ist nur dann sinnvoll, wenn der entsprechende Prädikattest kürzer dauert, als die Restlaufzeit der laufenden Transaktion. Sonst ist es günstiger die Transaktionen nacheinander laufen zu lassen.
Die Transaktionen müssen ebenfalls nacheinander laufen, wenn nicht garantiert werden kann, daß sie disjunkte Datenmengen benutzen, wenn also das schließlich getestete Prädikat erfüllbar ist. Da die Erfüllbarkeit des Testprädikates und zulange Testdauer zu demselben Synchronisationsverhalten führen, kann das Testverfahren erfüllbare Prädikate und Zweifelsfälle zusammenfassen zu:

> "Das Testverfahren nimmt an, daß das Prädikat erfüllbar ist."

Anders gesagt, das Testverfahren verlangt nur einseitige Korrektheit:

> Wenn der Test "unerfüllbar" liefert, muß das getestete Prädikat wirklich unerfüllbar sein.

Andere wichtige Anwendungen, z.B. die Anfrageoptimierung, fordern im Zweifelsfall dieselbe Entscheidung [Böttcher85].

## 2.2 Variation der Sup-Inf-Methode

Für den Test werden die Vergleiche jeder Konjunktion sortiert, sodaß ≠-Vergleiche zuletzt abgearbeitet werden. Außer ≠-Vergleichen werden alle Vergleiche wie bei der Sup-Inf-Methode durch folgende Regeln gerichtet:

```
                    a = b    ==>   ( a <= b ) AND ( b <= a )
                    c < d    ==>   c <= d + -1
                    e >= f   ==>   f <= e
                    g > h    ==>   h <= g + -1
Konstante <= x + offset      ==>   0 <= x + (offset-Konstante)
          x <= Konstante     ==>   x <= 0 + Konstante
```

Aus jeder gerichteten Konjunktion wird ein gerichteter Graph konstruiert: Die Konstante 0 sowie jedes Paar t.ai bilden je einen Knoten. Für jeden gerichteten Vergleich a <= b + Konstante wird eine Kante von a nach b gezogen und mit der Konstante bewertet. Enthält der so entstandene Graph einen Zyklus mit negativem Gewicht, so ist die Konjunktion unerfüllbar. Diese Zyklensuche hat für n Vergleiche den Aufwand (n hoch 3), wenn mit jeder Kante alle transitiven Kanten gezogen werden.

Im Unterschied zur Sup-Inf-Methode werden die ≠-Vergleiche nur dann in den Test einbezogen, wenn der aus allen anderen Vergleichen aufgebaute Graph keinen Zyklus mit negativem Gewicht enthält. Dabei ist ein Kompromiß zwischen dem Testzeitaufwand und der Menge der erzeugten Zweifelfälle zu schließen. Deshalb beschränkt sich der Konjunktionstest darauf, die Verträglichkeit jedes einzelnen ≠-Vergleichs mit dem aufgebauten Graphen zu testen:

> Ein Vergleich a ≠ b ist unverträglich, die untersuchte Konjunktion ist also unerfüllbar, falls es im Graphen Kanten a <= b und b <= a gibt.
> Entsprechend ist ein Vergleich c ≠ Konstante unverträglich, falls es im Graphen Kanten c <= 0 + Konstante und 0 <= c - Konstante gibt. Dann ist die Konjunktion ebenfalls unerfüllbar.

Sonst wird angenommen, daß die Konjunktion erfüllbar ist.

Die Zweifelsfälle dieses Verfahrens sind nur solche Konjunktionen, bei denen eine k-Clique (k>2) von durch ≠-Vergleiche verbundenen Variablen in einem zu kleinen Konstantenintervall liegen müßte. Solche Prädikate dürften in der Praxis äußerst selten formuliert werden. Damit ergibt sich für praktisch relevante Konjunktionen eine Testzeitkomplexität O(Vergleichsanzahl hoch 3).

## 3 Test beliebiger Prädikate

In diesem Abschnitt wird der Matrixtest zu einem Verfahren erweitert, das auf beliebige Prädikate anwendbar ist. Durch geeignete Umformungsschritte [Böttcher85] erzeugt man aus einem beliebigen Prädikat ein Prädikat in disjunktiver Pränex-Normalform (DPNF), das keine leeren Relationen enthält. Für den häufigen Fall, daß das Prädikat nur SOME-Variablen enthält, genügt der Matrixtest. Deshalb wird jetzt ein Testverfahren für Prädikate entwickelt, die ALL-quantifizierte Tupelvariablen ( ALL-Variablen ) enthalten. Der Grundgedanke ist, daß ein Prädikat unerfüllbar ist, wenn man durch Substitution der ALL-Variablen ein Gegenbeispiel findet. Deshalb wird ein Algorithmus entwickelt, der systematisch potentielle Gegenbeispiele erzeugt. Dieser ergibt zusammen mit dem Matrixtest ein Testverfahren für beliebige Prädikate.

### 3.1 Das Beweisprinzip

Um die Unerfüllbarkeit eines Prädikates P zu zeigen, konstruiert man ein potentielles Gegenbeispiel, indem man alle ALL-Variablen durch Konstanten substituiert. Das erhaltene Prädikat P' ist logische Konsequenz aus P ( P ==> P' ). Da P' keine ALL-Variablen enthält, kann es durch den Matrixtest getestet werden. Liefert der Matrixtest "P' ist unerfüllbar" ( P' ==> FALSE ), dann gilt
P ==> P' ==> FALSE , also P ist unerfüllbar.

Wendet man zum Beispiel auf das Prädikat P1

SOME t IN R ALL x IN R ( (t.a1 ≠ x.a1) AND (x.a2 < 20) )

die Substitution x:=t an, erhält man das Prädikat P1'

SOME t IN R ( (t.a1 ≠ t.a1) AND (t.a2 < 20) ) .

Der Matrixtest zeigt, daß P1' unerfüllbar ist, also ist P1 unerfüllbar. P1' ist das gesuchte Gegenbeispiel.

## 3.2 Zulässige Tupelsubstitutionen

Der folgende Aufzählalgorithmus für potentielle Gegenbeispiele benutzt eine Alternative zur Skolemisierung. Dafür unterscheidet er zwei Arten von Präzedenzen zwischen Tupelvariablen : Durch die Prädikatstruktur gegebene Präzedenzen ( < ), und durch Substitution hinzufügbare Präzedenzen ( <-- ).

Zwei Tupelvariablen eines Prädikates P heißen in Baumordnung ( t1 < t2 ), wenn in einem äquivalenten Prädikat AP in Antipränexform t2 im Gültigkeitsbereich von t1 definiert ist. Zum Beispiel hat Prädikat P1 die Baumordnungspräzedenz t < x.

Eine Tupelsubstitution eines Prädikates ist eine Funktion der Form

{ t1 <-- x1 , ..., tn <-- xn } ,

die alle ALL-Variablen xi entweder auf SOME-Variablen oder auf Sortenkonstanten cR **von derselben Sorte** wie xi abbilden. Zum oben genannten Prädikat P1 gibt es zwei Tupelsubstitutionen

T1 = { t <-- x } und T2 = { cR <-- x } .

Je mehr ALL-Variablen substituiert werden, desto wahrscheinlicher wird es, eine Unerfüllbarkeit zu finden. Deshalb wird die "Tupelsubstitution" so definiert, daß alle ALL-Variablen substituiert werden. ( Nach oben genannten Umformungen darf für jede Relation R die Existenz einer Sortenkonstante cR (eines Tupels), nicht jedoch die Existenz zweier verschiedener Sortenkonstanten vorausgesetzt werden. )

Substitutionen sind nicht immer mit der Baumordnung verträglich, wie ein Vergleich von Prädikat P1 mit folgendem Prädikat P2 zeigt :

ALL x IN R SOME t IN R ( (t.a1 ≠ x.a1) AND (x.a2 < 20) ) .

Hier ist die Festlegungspräzedenz x < t ( x muß vor t bekannt sein ) durch die Baumordnung der Tupelvariablen x und t vorgegeben. Die Substitution x:=t ( t <-- x ) ist für P2 nicht zulässig, denn sie würde zusammen mit der Festlegungspräzedenz x < t die zyklische Festlegungspräzedenz x < t <-- x ( x vor t vor x ) verlangen.

Um zulässige von unzulässigen Tupelsubstitutionen unterscheiden zu können, führen wir einen Präfixgraphen ein : Die Tupelvariablen und Sortenkonstanten bilden die Knoten, und gegebene Präzedenzen und Substitutionspräzedenzen bilden die Kanten. Wenn dieser Präfixgraph zyklenfrei ist, nennt man die Tupelsubstitution mit der Baumordnung verträglich oder kurz zulässig. Beispiel : Prädikat 3 :

ALL x IN R  ALL y IN S  SOME t IN R ( ... )

Gegeben sind die Präzedenzen { x < t , y < t } . Tupelsubstitutionen Ti zu Prädikat P3 sind :

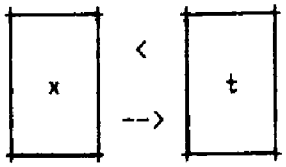
T1 = { t <-- x , t <-- y }  ( unzulässig ) ,

T2 = { t <-- x , cR <-- y }  ( unzulässig ) ,

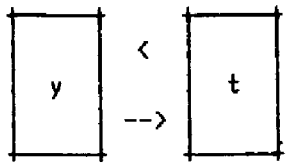
T3 = { cR <-- x , t <-- y }  ( unzulässig ) ,

T4 = { cR <-- x , cR <-- y }  ( zulässig ) .

Die Tupelsubstitutionen T1 und T2 führen mit der Baumordnung zum Zyklus:




Die Substitutionen T1 und T3 führen zum Zyklus:




Also ist nur die Tupelsubstitution T4 zulässig.

Der folgende wichtige Schluß gilt nur für zulässige Tupelsubstitutionen :
Kann man auf ein Prädikat eine zulässige Tupelsubstitution anwenden, und erhält man dabei ein unerfüllbares Prädikat, so war das untersuchte Prädikat unerfüllbar.
Diese Überlegung führt, verknüpft mit dem Matrixtest zu folgendem Theorem:

> Ein Prädikat in DPNF ist unerfüllbar,
> wenn es einen zyklenfreien Präfixgraphen gibt,
> sodaß (nach der Substitution) jeder Konjunktions-
> graph einen Zyklus mit negativem Gewicht enthält.

Aus diesem Theorem läßt sich folgender Testalgorithmus ableiten:

```
FOR EACH  Tupelsubstitution  DO
    IF  Präfixgraph ist zyklenfrei  THEN
        IF      der Matrixtest liefert, daß das substituierte
                Prädikat unerfüllbar ist  THEN
            RETURN  ("Das Prädikat ist unerfüllbar")
        END ;
    END ;
END ;
RETURN  ("Es wird angenommen, daß das Prädikat erfüllbar ist") ;
```

Der Zyklentest im Präfixgraphen läßt sich mit demselben Algorithmus wie der Test im Konjunktionengraphen durchführen, wenn man den Präfix wie eine Konjunktion testet und für Baumordungs- und Tupelsubstitutionskanten ( < und <-- ) <-Vergleiche setzt [Böttcher85].

Die folgende Verbesserung dieses Algorithmus setzt voraus, daß im Präfixgraphen zu neuen Baumordnungs- oder Substitutionspräzedenzen immer alle transitiven Kanten gezogen werden. Somit ist eine neue Präzedenz a <-- b genau dann zulässig, wenn der Graph keine Präzedenz b <-- a enthält.

### 3.3 Beschränkung der Matrixtests auf maximale Äquivalenzrelationen

Die Einführung der Begriffe "Äquivalenzrelation" [Bibel82] und "maximal" ermöglicht es, einen Testalgorithmus zu formulieren, der mit weniger Matrixtests die gleichen Ergebnisse liefert. Es genügt nämlich, die Tupelsubstitutionen zu testen, die "maximal" viele Tupelvariablen gleichsetzen.

Jede Tupelsubstitution erzeugt eine Äquivalenzrelation "==", die angibt, welche Tupelvariablen nach der Substitution gleich sind :
Wird xi durch eine SOME-Variable ti ersetzt, gehören beide zu derselben Äquivalenzklasse, also ti == xi . Werden ALL-Variablen x1 und x2 durch dieselbe Sortenkonstante cR substituiert, gehören beide zu derselben Äquivalenzklasse, also x1 == x2 .
Um anschließend den Begriff maximal definieren zu können, ist es sinnvoll, Sortenkonstanten cR aus den Äquivalenzklassen auszuschließen, die Äquivalenzrelation == also wie folgt zu definieren:

1. Jede SOME-Variable ti bildet eine eigene Äquivalenzklasse mit dem Namen ti und enthält auch ti.
2. Jede Relation R bildet eine eigene Äquivalenzklasse mit dem Namen cR.
3. Jede Substitution ti <-- xi ordnet der Äquivalenzklasse mit dem Namen ti die ALL-Variable xi zu.

Anmerkungen :

1. Elemente **derselben** Äquivalenzklasse **müssen** gleich sein, Elemente **verschiedener** Äquivalenzklassen **können** gleich sein! Deshalb bildet jede SOME-Variable eine eigene Äquivalenzklasse.
2. Da Tupelsubstitutionen und Äquivalenzrelationen in 1:1-Beziehung stehen, wird zwischen beiden Begriffen nicht weiter unterschieden, auch nicht in Verbindung mit den Begriffen "zulässig" und "maximal".

"Maximal" sind solche Äquivalenzrelationen, in denen soviele Tupelvariablen wie möglich gleichgesetzt, d.h. soviele Äquivalenzklassen wie zulässig vereinigt werden :

Eine Äquivalenzrelation heißt maximal, wenn

1. sie zulässig ist **und**
2. für jede Vereinigung zweier nicht leerer Äquivalenzklassen derselben Sorte (Relation) gilt :
   Sie enthält zwei SOME-Variablen **oder**
   die neue Äquivalenzrelation ist nicht mit der Baumordnung verträglich.

Anmerkungen :

1. Die Vereinigung mit einer leeren cR-Klasse wird ausgeschlossen, weil dies nicht zu mehr Gleichsetzungen führt.
2. Die Vereinigung zweier Äquivalenzklassen mit SOME-Variablen ist auszuschließen, da SOME-Variablen nicht gleich sein müssen, ihre Gleichsetzung also nicht erzwungen werden darf.

Beispiel : Zum Prädikat

```
SOME t IN R   ALL x IN R   ALL y IN R   ( ... )
```

gehören die Tupelsubstitutionen und die Äquivalenzrelationen :

```
T1 = {  t <-- x ,  t <-- y }      ÄR1 = { [ t == x == y ] , [ ] }   ( maximal )

T2 = {  t <-- x , cR <-- y }      ÄR2 = { [ t == x ] ,   [ y ] }

T3 = { cR <-- x ,  t <-- y }      ÄR3 = { [ t == y ] ,   [ x ] }

T4 = { cR <-- x , cR <-- y }      ÄR4 = { [ t ] ,   [ x == y ] }
```

Die Äquivalenzrelationen ÄR2 bis ÄR4 sind nicht maximal, denn immer können ihre beiden Äquivalenzklassen zulässig zu [ t == x == y ] vereinigt werden.

Matrixtests unter nicht maximalen Äquivalenzrelationen durchzuführen ist überflüssig, wie die folgende Überlegung zeigt :
Nehmen wir an, in einer Äquivalenzrelation eines Prädikates

... SOME t IN R ... ALL x IN R ( P( t, x, ... ) )

ist die Gleichsetzung t == x zulässig. Dann ist es überflüssig, den Matrixtest das Prädikat

... SOME t IN R ... SOME x IN R ( P( t, x, ... ) )

testen zu lassen - der Test des maximalen Prädikates

... SOME t IN R ... ( P( t, t, ... ) )

genügt. Denn aus

... SOME t IN R ... ( P( t, t, ... ) )

==> ... SOME t IN R ... SOME x IN R ( P( t, x, ... ) )

folgt : Findet der Matrixtest unter dem nicht maximalen Prädikat einen Widerspruch,

... SOME t IN R ... SOME x IN R ( P( t, x, ... ) ) ==> FALSE

gilt auch

... SOME t IN R ... ( P( t, t, ... ) ) ==> FALSE .

Weiterhin ist der Matrixtest so angelegt, daß er die Unerfüllbarkeit unter der maximalen Äquivalenzrelation finden muß. Daraus ergibt sich folgender Satz :

Satz 1

Jede Unerfüllbarkeit, die der Matrixtest unter Tupelsubstitutionen findet, findet er auch, wenn er ausschließlich unter maximalen Tupelsubstitutionen testet.

Also genügt es, Matrixtests unter maximalen Tupelsubstitutionen durchzuführen.

Im letzten Beispiel genügt es, den Matrixtest unter der Äquivalenzrelation ÄR1 durchzuführen. Denn, angenommen der Matrixtest findet eine Unerfüllbarkeit unter ÄR2, d.h. nach Gleichsetzung von t und x, y jedoch bleibt beliebig; dann muß er unter der umfassenderen Äquivalenzrelation { [ t == x == y ] } auch eine Unerfüllbarkeit finden, wenn zusätzlich zu t = x noch t = y gefordert wird.

### 3.4 Ein Maximalitätstest

Wie testet man, ob eine zulässige Äquivalenzrelation maximal ist? Sie ist dann maximal, wenn jede Vereinigung einer nicht-leeren cR-Klasse mit einer SOME-Klasse ti derselben Sorte zu einem Zyklus im Präfixgraphen führt.
Die Vereinigung einer cR-Klasse [ x1, ..., xn ] mit einer Klasse ti = [ ti, y1, ..., yn ] fügt zum Präfixgraphen neue Substitutionskanten ti <-- x1, ..., ti <-- xn hinzu. Eine dieser Kanten ti <-- xj führt genau dann zum Zyklus, wenn es im Graphen schon eine Kante xj <-- ti gibt, also wenn es eine Kante cR <-- ti gibt. Dies genügt, denn zu cR <-- xj und xj <-- ti muß es im Präfixgraphen auch eine transitive Kante cR <-- ti geben.

Erweitert man diese Überlegung auf alle Paare von Klassen, ergibt das den Satz :

Satz 2

Eine zulässige Äquivalenzrelation ist genau dann maximal, wenn für jede SOME-Klasse ti gilt :
Der Präfixgraph enthält eine Präzedenz cR <-- ti zur cR-Klasse derselben Sorte **oder** die cR-Klasse ist leer.

Eine cR-Klasse ist **leer**, wenn es keine Präzedenz
cR <-- Tupelvariable gibt.

Der Maximalitätstest läßt sich also vollständig auf die Suche nach bestimmten Kanten im Präfixgraphen reduzieren. Der Suchaufwand ist der Anzahl der SOME-Variablen proportional und damit wesentlich geringer als der Aufwand der Matrixtests.

### 3.5 Der Testalgorithmus

Um alle Tupelsubstitutionen aufzuzählen, muß man alle Zuordnungskombinationen von ALL-Variablen zu Klassen ihrer Relation erzeugen.
Der dafür implementierte Backtracking-Algorithmus schließt einen Zulässigkeitstest mit ein. Ausgehend von der Baumordnung werden nur zyklenfreie Präfixgraphen aufgebaut :

```
Übertrage die Baumordnungspräzedenzen und ihre transitiven Kanten
in den Präfixgraphen, und markiere alle Kanten mit 0 ;
Sortiere bzw. numeriere die n ALL-Variablen : x1 , ... , xn ;
Substituiere ALL-Variablen ab Nr( 1 ) ;
RETURN ("Es wird angenommen, daß das Prädikat erfüllbar ist") ;


Kern des Algorithmus ist folgende rekursive Prozedur :

Substituiere ALL-Variablen ab Nr( i ) :

IF  alle ALL-Variablen sind  zugeordnet ( i > n )  THEN
    ( die Tupelsubstitution T ist zulässig )
    IF  die Tupelsubstitution ist auch maximal  THEN
        IF      der Matrixtest liefert, daß das substituierte
                Prädikat PT unerfüllbar ist  THEN
            EXIT-RETURN ("das untersuchte Prädikat P ist unerfüllbar")
        END
    END
ELSE
    FOR EACH  Sorten respektierende Substitution für xi  DO
        IF  diese Substitution ist zulässig  THEN
            Speichere sie und ihre transitiven Kanten mit Markierung i ;

            Substituiere ALL-Variablen ab Nr( i+1 ) ;

            Lösche mit i markierte Kante aus dem Präfixgraphen
        END
    END
END ;
```

### 3.6 Aufwandsabschätzung und Alternativen

Die Anzahl der Tupelsubstitutionen ¦ T ¦ eines Prädikates hängt primär von der Quantorenzahl pro Relation ab:

¦ T ¦ = P r o d u k t von ( ( SOMEs + 1 ) hoch ALLs )
s=1..Relationenanzahl

SOMEs = Anzahl der SOME-Variablen gebunden an Relation s ;
ALLs = Anzahl der ALL-Variablen gebunden an Relation s .

Eine früh gefundene Unzulässigkeit verringert die Anzahl der erzeugten Tupelsubstitutionen beträchtlich. Dennoch gibt es Prädikate, in denen alle Tupelsubstitutionen zulässig sind.
Weitaus kleiner als die Zahl der zulässigen Tupelsubstitutionen ist die Anzahl der maximalen Tupelsubstitutionen. Eine Formel für die obere Grenze ist nicht bekannt.

Der Aufwand für den Aufbau des Präfixgraphen ist höchstens O( ( Quantorenzahl + Relationenzahl ) hoch 3 ) . Der Engpaß des Verfahrens ist im allgemeinen der Matrixtest mit dem Aufwand O( Vergleichsanzahl hoch 3 ).

Nach Abschätzung des Testaufwands und des erwarteten Effizienzgewinns bei positivem Testergebnis bleiben dem System folgende Alternativen zum erweiterten Matrixtest :

1. Das Prädikat wird sofort wie ein Zweifelsfall behandelt, die Transaktionen laufen also nacheinander.
2. Das System fordert für einzelne Transaktionen einfacher testbare Obermengen der benötigten Daten an.
3. Das System prüft, ob ein anderes, z.B. auf Resolution basierendes Verfahren schneller ein Ergebnis erwarten läßt.

## 4 Zusammenfassung

Transaktionen dürfen parallel ablaufen, wenn das Datenbanksystem garantieren kann, daß die benutzten Datenmengen disjunkt sind.

Durch die Alternative, Transaktionen nacheinander ablaufen lassen zu können, ist ein dafür entwickeltes Testverfahren nicht in jedem Einzelfall zu einer Entscheidung gezwungen. Stattdessen muß die erforderliche Testzeit kurz und im voraus abschätzbar sein. Deshalb behandelt der erweiterte Matrixtest (in der Praxis seltene) zu komplizierte Prädikate als Zweifelsfälle, also so, daß das Datenbanksystem die Transaktionen nacheinander ablaufen läßt.

Dem erweiterten Matrixtest liegt folgendes Beweisverfahren zugrunde :
Ein Prädikat in DPNF ist unerfüllbar, wenn es einen zyklenfreien Präfixgraphen gibt, sodaß jeder Konjunktionsgraph einen Zyklus mit negativem Gewicht hat.

Die Effizienz des erweiterten Matrixtests kann mit einfachen Maximalitätstests noch gesteigert werden, denn es gilt :
Matrixtests können auf maximale Äquivalenzrelationen beschränkt werden, ohne daß das Testverfahren dadurch unvollständiger wird.

Der erweiterte Matrixtest ist zwar beschränkt auf eine Teilklasse von Prädikaten 1.Ordnung, jedoch ist er, wie die Aufwandsabschätzung zeigt, für die häufigen Prädikate mit wenigen Tupelvariablen pro Relation gut geeignet. Dies bestätigt auch die Implementation.

**Danksagung:** Diese Arbeit wurde zum Teil von der deutschen Forschungsgemeinschaft gefördert unter der Projektnummer SCHM450/3-1 ( Leitung Prof. J.W. Schmidt ). Herrn Dr. Volker Linnemann danke ich herzlich für Anregungen beim Lesen früherer Versionen dieser Arbeit.

## Literatur

[Andrews81]
Andrews, P.B.: Theorem Proving via General Matings.
JACM 28, 2, 1981. pp. 193-214.

[Appelrath85]
Appelrath, H.J., Bense, H.: Zwei Schritte zur Verbesserung von PROLOG-Programmiersystemen : DB-Unterstützung und Meta-Interpreter.
In Blaser, A., Pistor, P. (Eds.): Datenbank-Systeme für Büro, Technik und Wissenschaft. Berlin, Springer, 1985. pp. 161-177.

[Bibel81]
Bibel, W.: On Matrices with Connections. JACM 28, 4, 1981. pp. 633-645.

[Bibel82]
Bibel, W.: Automated Theorem Proving. Braunschweig, Vieweg, 1982.

[Bläsius81]
Bläsius, K., Eisinger, N., Siekmann, J., Smolka, G., Herold, A., Walther, C.: The Markgraf Karl Refutation Procedure. Proceedings 7th IJCAI, Vancouver, 1981.

[Bledsoe77]
Bledsoe, W.W.: Non-resolution Theorem Proving.
Artificial Intelligence 9 , 1977, pp. 1-35.

[Böttcher85]
Böttcher, S.: Ein Testverfahren für Datenbankprädikate.
Bericht Nr. 114, Universität Hamburg, Fachbereich Informatik, 1985.

[Brodie84]
Brodie, M.L., Mylopoulos, J.L., Schmidt, J.W. (Eds.): On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages.
Berlin [u.a.], Springer, 1984.

[Codd72]
Codd, E.F.: Relational Completeness of Data Base Sublanguages.
Courant Computer Science Symposia 6, pp. 65-101.

[Edelmann84]
Edelmann, J.: Die Datenbankprogrammiersprache DBPL:
Eine Beschreibung ausgewählter Sprachkonstrukte und deren Implementation.
Diplomarbeit, Universität Hamburg, Fachbereich Informatik, 1984.

[Eswaran76]
Eswaran, K.P., Gray, J.N., Lorie, R.A. Traiger, I.L.: The Notions of Consistency and Predicate Locks in a Database System.
CACM, Vol.19, No.11, Nov. 1976, pp. 624-633.

[Garey79]
Garey, M.R., Johnson, D.S.: Computers and Intractability.
Bell Telephon Laboratories, 1979.

[Hsiang83]
Hsiang, J., Dershowitz, N.: Rewrite Methods for Clausal and Non-Clausal Theorem Proving. Automata, Languages and Programming, 10th Colloquium, Barcelona, Juli 1983, Berlin [u.a.], Springer 1983.

[Hunt79]
Hunt, H.B., Rosenkrantz, D.J.: The Complexity of Testing Predicate Locks.
Proceedings ACM-SIGMOD International Conference on Management of Data, Mai 1979. pp. 127-133.

[Kleene71]
Kleene, S.C.: Introduction to Metamathematics. Wolthers-Noordhoff, North Holland, 1971.

[Klug83]
Klug, A.: Locking Expressions for Increased Database Concurrency. JACM 30, 1, Januar 1983. pp. 36-54.

[Kowalski75]
Kowalski, R.: A Proof Procedure Using Connection Graphs. JACM 22, 4, Oktober 1975. pp. 572-595.

[Mall84]
Mall, M., Reimer, M., Schmidt, J.W.: Data Selection, Sharing and Access Control in a Relational Scenario. In [Brodie84].

[Munz79]
Munz, R., Schneider, H.J., Steyer, F.: Application of Sub-Predicate Tests in Database Systems. Proc. of 5th International Conference on Very Large Data Bases, Rio de Janeiro, October 1979.

[Reiter84]
Reiter, R.: Towards a Logical Reconstruction of Relational Database Theory. In [Brodie84].

[Robinson65]
Robinson, J.A.: A Machine Oriented Logic Based on the Resolution Principle. JACM 12, 1, Jan. 1965. pp. 23-41.

[Rosenkrantz80]
Rosenkrantz, D.J., Hunt, H.B.: Processing Conjunctive Predicates and Queries. Proc. of 6th International Conference on Very Large Data Bases, Montreal, October 1980. pp.64-74.

[Schmidt77]
Schmidt, J.W.: Some High Level Language Constructs for Data of Type Relation. ACM Transactions on Database Systems 2, 3, September 1977.

[Shostak79]
Shostak, R.E.: Deciding Linear Inequalities by computing loop residues. In Joyner, W.S. (Ed.): Proc. 4th Workshop on Automated Deduction, Austin, Texas, 1979, pp. 81-89.