LMU – Proseminar „Grundlagen höherer Programmiersprachen"

Thema: „Wie ein beispielhaftes Query-System funktioniert"

Vortragender: Petar Tonev

1. Vorworte

-
- ***Ziele und Aufgaben des Vortrags*** – klären wie das für Beispiel verwendete Anfragesystem die Herleitung musterbasierter Informationen aus einer Datenbank ermöglicht
- ***Grundlagen des Anfragesystems*** – Mustervergleich und Unifikation
- ***Implementierung und Beispiele***: durch die Query-Sprache aus dem Buch „Structure and Interpretation of Computer Languages"

---

2. Mustervergleich (Pattern Matching)

Syntax eines Datums: ***(*** *(<Wert>*)* (Wert)* ***)***
Beispiele: (a), (a a), ((2 a) b), ((b a) 7 ((a c) d)), . . .

Syntax eines Musters: ***(*** *(<Variable> <Variable>* <Wert>*)** ***)***
Beispiele : (a ?x), (?x ?y), ((?x a) 7 ((a ?y) d))

-ein ***Mustervergleicher*** ist ein Programm, das überprüft, ob ein Datum zu einem gegebenen Muster passt.

Beispiel: das Datum **((a b) c (a b))** entspricht
-dem Muster **(?x c ?x)**, wenn **?x←(a b)**;
-dem Muster **(?x ?y ?z)**, wenn **?x←(a b)** , **?z←(a b)** , **?y←c**
-dem Muster **((?x ?y) c (?x ?y)),** wenn **. . . ?**
Das Datum entspricht nicht dem Muster **(?x a ?y)** *(Warum?)*

-Eingabe und Ausgabe eines Mustervergleichers – Muster, Datum, Bindungsrahmen

*Bindungsrahmen – assoziatives Array, wo die Schlüsselvariablen Namen sind
Syntax: *[<Variablenname**1**>←<Wert **a**>; . . . <Variablenname **n**>←<Wert **t**>]*
Beispiel :

?x	**A**
?y	**12**
?z	**John**

Der Mustervergleicher überprüft ob das Datum dem Muster in einer Weise entspricht, die mit den Bindungen aus dem Rahmen konsistent ist.

-falls ja, werden der Rahmen + eventuelle neue Bindungen abgeliefert.

-falls nein, Angabe „Vergleich nicht erfolgreich“

Beispiel: Muster **(?x ?y ?x)**, Datum **(a b a)** ,
-bei leerem Bindungsrahmen, =>
Ausgabe :

?x	**a**
?y	**b**

-bei Eingaberahmen

?y	**a**
?x	**?**

=>Fehlschlag (Vergleich nicht erfolgreich)
-bei Eingaberahmen

?y	**b**
?x	**?**

Ausgabe :

?y	**...?**
?x	**...?**

---

3. Ströme von Bindungsrahmen

Problem: Darstellung aller Ergebnisse eines Vergleichs

→ein ***Datenstrom*** ist eine „verzögerte“ Liste , in Scheme etwa:

(cons <*a*> (delay <*b*>)), was gleich zu **(cons-stream <*a*> <*b*>)** ist.

delay ist Sonderform , die den Term <**b**> nicht auswertet, sondern ein sog. *verzögertes Objekt* liefert, das später ausgewertet werden soll. Dies erfolgt durch die Prozedur ***force*:**

```
(define (stream-car.stream) (car stream))
(define (stream-cdr.stream) (force (cdr stream)))
```

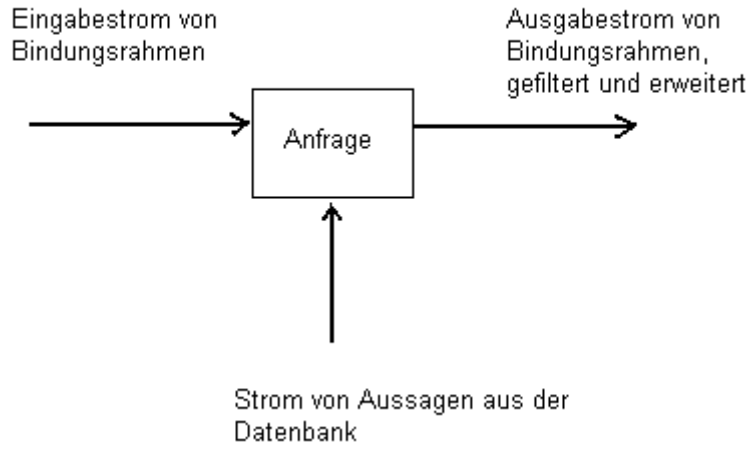
/ Zunächst das erste Element des Stromes auswerten; erst bei Anforderung durch **force** (das heißt „forciere“) auch den **tail** des Stromes auswerten. */*

→***Strom von Bindungsrahmen*** - Liste von Bindungen, manche von denen noch dazukommen können

Beispiel für Strom von Bindungsrahmen:
{(?x←2; ?y←5) (?z←?x) (?z←?) …}

→bei Eingabe eines Rahmens werden die Einträge der Datenbank durchlaufen
→für jeden Eintrag: Erweiterung für den Rahmen oder spezielles Symbol fürs Fehlschlagen
→alle Ergebnisse werden zu einem Datenstrom zusammengefasst.
→der Ergebnisstrom wird durch Filter geschickt, um Fehlschläge auszusortieren.

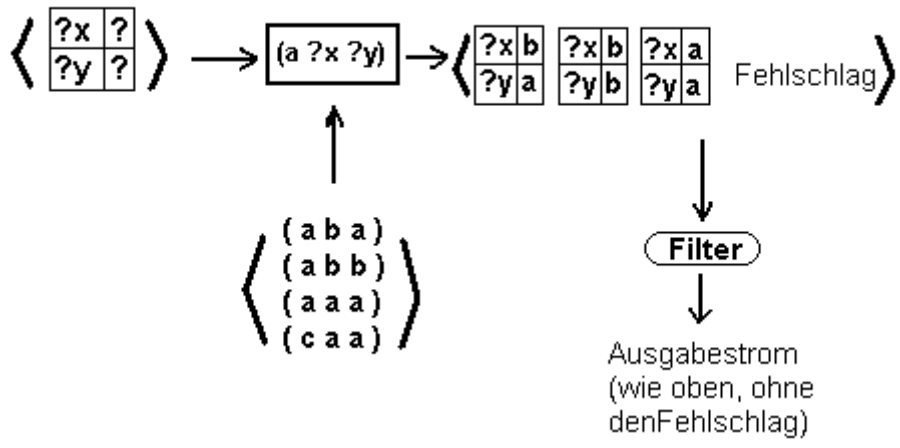
Abbildung:*Das Anfragesystem, beschrieben durch Datenströmen*



→bei **einfachen Einfragen** besteht der Eingabestrom aus einzigem leeren Bindungsrahmen, der Ausgabestrom – aus alle Erweiterungen des leeren Rahmens.
D.h. die Ausgabe ist Kopie des Anfragemusters mit instantiierten Variablen.

Syntax von einfachen Anfragen: *(<Variable>* <Konstante>*)*
z.B. =>(job ?x ?y)
(job A a)
(job B b)
(job C b)

Abbildung: *Das Anfragesystem mit beispielhaften Eingabeströmen, Datenkbank und Anfrage*



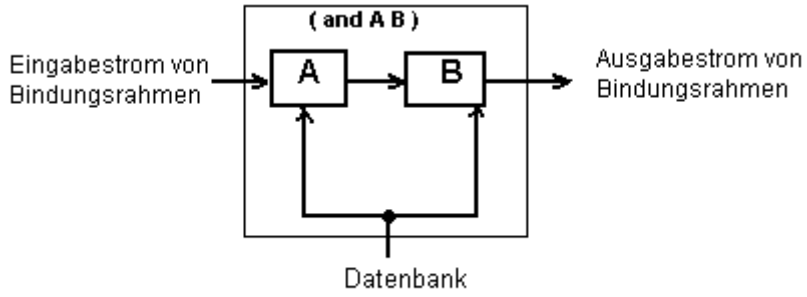
4. Zusammengesetzte Anfragen

4.1. **and**

Beispiel: (and (kann-taetigkeit ?x (computer programmierer trainee))
(taetigkeit ?person ?x))

-zuerst finde die Einträge, die dem ersten Muster entsprechen, dann dem zweiten (sequenziell)
-die Rahmen, die der erste Anfragefilter durchlässt werden von der zweiten Anfrage noch mal gefiltert und erweitert

Abbildung: *and* *von zwei Anfragen A und B*

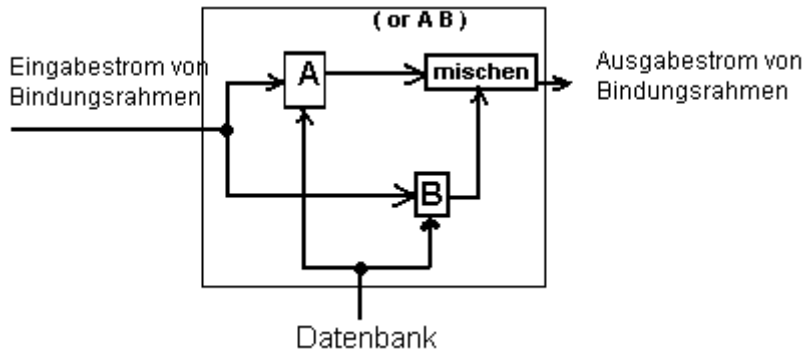


4.2. or

Beispiel: (or (kann-taetigkeit ?x (computer programmierer trainee))
(kann-taetigkeit ?x (computer programmierer)))

-der eingegebene Strom von Bindungsahmen wird für jede Anfrage getrennt erweitert (parallel)

Abbildung: *or* *von zwei Anfragen A und B*



4.3. not

-„not“ verhält sich wie ein Filter, der Bindungsrahmen entfernt, die der Anfrage entsprechen.

Beispiel:

(not (taetigkeit ?x (computer programierer)))

Ergebnis: Strom aus Rahmen, in denen die Bindung für die Variable **?x** dem Anfragemuster nicht entspricht

4.4. lisp-value

Beispiel: (and (gehalt ?person ?betrag)
(lisp-value > ?betrag 50000))

-„lisp-value“verhält sich auch wie ein Filter
-das lisp-Prädikat entfernt aus dem Eingabestrom alle Rahmen, für die es fehlschlägt

---

5. Unifikation

5.1. Substitution, Instanz, Unifikator

→***Substitution***: Abbildung von der Menge der Variablen in die Menge der Terme **ϑ:√→T(Σ)**, die nur an endlich vielen Stellen von der Identität abweicht.
(die Identität ist die leere Substitution).

→***Instanz*** :eineFormel φ ist Instanz einer Formel **Ψ**, falls es eine Substitution **ϑ** gibt mit **φ =Ψϑ** (Postfixnotation)

→eine Substitution heißt ***Unifikator*** für die Menge **S = {A1,A2,...An}**, falls alle Instanzen der Formel **A1,...An** gleich sind
(A1ϑ = ...=Anϑ)

-Gibt es Unifikator =>**S** *unifizierbar.*
-allgemeinster Unifikator ϑ - falls es für jeden Unifikator **σ** aus **S** eine Substitution **β** gibt, sodass **σ = ϑβ**

Beispiele:

1) **{ f(x, y), f(3, z), f(3, 7)}**
ϑ = { x←3, y←7, z←7 }

2) **{f(x, y), f(3, z), f(5, 7) }**
nicht unifizierbar

3) **{f(x), f(y)}**
ϑ = { x ← y } *(allgemeinster Unifikator; z.B. {x← 3}, {y← 3} Unifikatoren, aber nicht allgemeinst)*

5.2. Unifikation im Anfragesystem

♦ **Unifikation** - Verallgemeinerung des Mustervergleichs, wo das Datum ***Variablen enthalten kann.***
Wozu braucht man Variablen im Datum? → Möglichkeit Regeln anzuwenden

♦ein ***Unifikationsagorithmus*** stellt für zwei Aussagen fest, ob es möglich ist, an die Variablen Werte zuzuweisen, sodass die Aussagen gleich werden.

Beispiele:

(1) **⇒(?x a ?y)**
(?y ?z a)

Ausgabe:

?x	**a**
?y	**A**
?z	**A**

(2) **⇒(?x ?y a)**
(?x b ?y)

Ausgabe: Unifikation erfolglos *(keinen Wert für y, mit dem die Muster gleich)*

♦**Ableitungen notwendig** (bei komplexen Mustern)

z.B. unifiziere **(?x ?x)** und **((a ?y c) (a b ?z))**

Der Unifikationsalgorithmus muss ableiten, dass
?x←(a b c), ?y←b, ?z←c
Lösung: Gleichungen zwischen den Musterkomponenten.
Hier im Beispiel:

?x = (a ?y c)
?x = (a b ?z) **=> (a ?y c) = (a b ?z) => a = a, ?y = b, c = ?z => ?x = (a b c)**

/****Bemerkung****:* manche Variablen können ungebunden bleiben, und andere an variablenenthaltenden Terme gebunden werden.

Beispiel: unifiziere **(?x a)** und **((b ?y) ?z)**
Ableitung: **?x = (b ?y)**
a = ?z

-?x und **?y** weiter nicht auflösbar .
- ?y nicht eingeschränkt => keine Bindung für **?y**.
-?x eingeschränkt durch **(b ?y) =>** Bindung für **?x**, die Variable beinhaltet) ***/**

6. Anwendung von Regeln

→Syntax von Regeln: (rule <*Folgerung*><*Rumpf*>)

→Verfahren:

(1) Unifiziere die Anfrage mit der Folgerung der Regel → Erweiterung für den (ursprünglichen) Rahmen

(2) Werte den Rumpf der Regel aus, relativ zu dem erweiterten Bindungsrahmen

Beispiel:
(wohnt-in-der-naehe ?x (Hacker Alyssa P))

anzuwendende Regel:

(rule ((wohnt-in-der-naehe ?person-1 ?person-2) *// Folgerung*
(and (adresse ?person-1 (stadt . ?rest-1)) *// Rumpf*
(adresse ?person-2 (stadt . ?rest-2))
(not (lisp-value equal? ?person-1 ?person-2)))) *//damit P1≠P2*

1) Unifikation ergibt: **?person-2←(Hacker Alyssa P)**
2) Relativ zu diesem Rahmen werte den Rumpf aus =>Erweiterung des Rahmens um eine Bildung für **?person-1** =>**?x** mit diesem Wert instantiiert.

Einfache (nicht zusammengesetzte) Anfragen

Abbildung: *Auswertung von einfachen Anfragen bei Anwendung von Regeln*